SOLUTION APPROACHES TO A MULTI-STAGE, MULTI-MACHINE, MULTI-PRODUCT PRODUCTION SCHEDULING PROBLEM

BY

CHARLES STAFFORD LOVELAND

A DISSERTATION PRESENTED TO THE GRADUATE COUNCIL OF
THE UNIVERSITY OF FLORIDA IN PARTIAL FULFILLMENT OF
THE REQUIREMENTS FOR THE DEGREE OF DOCTOR OF PHILOSOPHY

UNIVERSITY OF FLORIDA

1978

TO

MOM AND DAD

ACKNOWLEDGMENTS

I wish to acknowledge the considerable contributions to this work by Dr. Thom J. Hodgson, who acted as the Chairman of my Doctoral Committee. His encouragement, support, and periodic "pep talks" kept me from wavering from my course. His guidance and friendship are sincerely appreciated. The insight and experience of Dr. Hodgson contributed considerably to the form and content of this dissertation and to my education.

I thank the remainder of my committee, Dr. Richard L. Francis, Dr. Gary J. Koehler, Dr. Timothy J. Lowe, and Dr. H. Donald Ratliff for their constructive comments and suggestions leading to the completion of this work. I am also grateful to Dr. James F. Burns, Dr. Donald W. Hearn, Dr. Eginhard Muth, and Dr. Michael E. Thomas for their assistance and support throughout my stay at the University of Florida.

I also wish to thank Miss Adrian Agerton, Miss Cheryl Gray, Mr. Russell Plumb, and Miss Billie Jo Zirger for their many long hours of physical and artistic assistance in the many tasks in the preparation of this work.

Thanks are also extended to the office staff of the ISE Department for their aid and for putting up with my disruption of the normal office functions in those final hectic days.

I also acknowledge Mrs. Beth Beville for her excellent typing of the manuscript.

Finally, I thank and lovingly dedicate this dissertation to my parents, Warren and Barbara Loveland, for their constant understanding, encouragement, and faith in me throughout the course of this work and my other endeavors. No son could ask for more.

This dissertation was supported in part under ONR contract number N00014-76-C-0096.

TABLE OF CONTENTS

PAGE

LIST OF FIGURES

FIGURE		PAGE

LIST OF TABLES

Abstract of Dissertation Presented to the Graduate Council of the University of Florida in Partial Fulfillment of the Requirements for the Degree of Doctor of Philosophy

SOLUTION APPROACHES TO A MULTI-STAGE, MULTI-MACHINE, MULTI-PRODUCT PRODUCTION SCHEDULING PROBLEM

By

Charles Stafford Loveland

December 1978

Chairman: Dr. Thom J. Hodgson
Major Department: Industrial and Systems Engineering

Consider an M-product, N-stage (acyclic machine network), finite horizon scheduling problem. Demand for the products over H scheduling periods is known, but not necessarily constant. There are N_j identical machines at each stage j. The objective is to schedule this multi-machine system so as to minimize the sum of production and inventory costs over the scheduling horizon.

Two cases of the problem are investigated--the frontal bottleneck and the posterior bottleneck. Both are caused by the production rate of each product being a monotonic function of its stage of completion. It is shown that under certain conditions a single-pass algorithm provides optimal solutions to the frontal bottleneck case. For the posterior bottleneck an efficient heuristic is developed. Over 98% of the randomly generated test problems are solved optimally by the heuristic.

CHAPTER 1

INTRODUCTION AND LITERATURE REVIEW

1.1 Introduction

If one considers the structure of a production system, it is often the case that whatever is produced is processed through several stages of production. For instance, a gear blank for an automobile transmission is first turned and given an initial shape on a screw machine. The gear teeth are cut using a gear hobber. Then the gear goes through a series of machining operations which result in a finished gear, ready for assembly into a transmission. It is also many times the case that more than one product is produced on any given set of facilities (stage). In this situation it is necessary to schedule the products to be produced over the stages in order to satisfy the demand for the products. It is assumed (with little loss in generality) that time can be discretized for scheduling purposes into periods (i.e., shifts, days, weeks). It is also assumed that the demand for each product is known sufficiently well to be used for planning purposes over some horizon. It is this scenario that provides the setting for this dissertation.

The multi-stage production scheduling problem studied here is concerned with scheduling the production of a set of products on a set of machines which must perform their functions in a given order. Each of these functions is a stage in the completion of the product and each stage has a given production rate for each product. A machine can perform only one function; however, there may be more than one

machine at each stage. An in-process inventory is maintained at each stage to store the goods which have just completed this stage.

The system of stages can be depicted as a directed, acyclic network. The nodes are the stages. The arcs represent the direction of flow of unfinished products through the production system. If such a network has at most one incoming arc at each node, it is called an arborescence, as in nodes 1 through j-1 in Figure 1.1. If it has at most one outgoing arc at each node, it is called an assembly network, as in nodes 2 through j in Figure 1.1.

The objective is usually to schedule and determine lot sizes for the production of the different products while minimizing some function of the production costs, set-up costs, and inventory carrying costs. In some problems backlogging of demand is considered, which causes a shortage cost.

There exists terminology in the literature which it is convenient to adopt here. A model is deterministic when the external demand on the system is known in advance with certainty. In a stochastic model these demands are random variables having a probability distribution which is assumed to be known. A stationary model defines its external demands with parameters which are assumed to be independent of time. In a nonstationary model, these parameters may vary over time. Backlogging of demand allows unsatisfied demand to be satisfied by products completed at a later date. If no backlogging is allowed, sales from unsatisfied demand are lost.

1.2 Literature

This literature review covers the more recent papers on the multi-stage production scheduling problem with emphasis on the

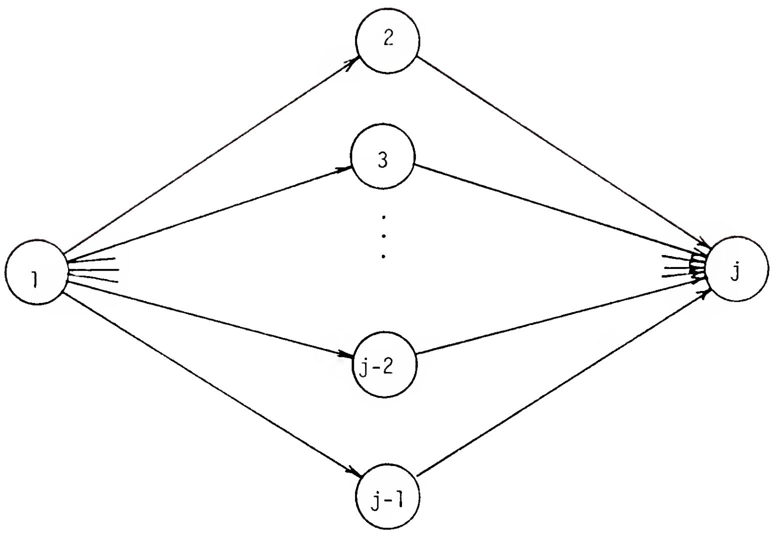

Figure 1.1. An example network.

deterministic models. In the process the problems treated by the papers are categorized in order to delineate the unique area into which this dissertation falls. The stochastic problems are basically inventory control problems and have no constraints on resources or on the number of machines used in any production processes. The deterministic problems fall into two main categories - finite and infinite horizon. With one exception both categories deal with a problem in which a single product is produced. Excluding Von Lanzenauer [44] among the finite horizon problems and Jensen and Khan [25] among the infinite horizon problems, all the deterministic problems assume an infinite resource supply or enough machines available at each stage to produce the desired batch size. The problem investigated in this dissertation is deterministic, has a finite scheduling horizon, deals with multiple products and constrains the number of machines at each stage. Only the Von Lanzenauer paper deals with all four of these properties.

The latest survey of the literature was done by Clark [6] and covers publications through 1971.

There are four major techniques which are generally employed to analyze the stochastic multi-stage problem: expected cost analysis, stationary process analysis, dynamic process analysis, and network theory.

Among the expected cost models, Berman and Clark [3] treated a single product in a two-level arborescence stage structure. Hadley and Whitin, [18] and [19], analyzed a single-level system of parallel stages whose demands are assumed to have an independent, stationary Poisson distribution. The same problem was investigated by Gross [17] considering a single time period. Krishnan and Rao [28] also considered the problem as formulated by Gross.

A stationary process analysis was used by Love [31] on a two-stages-in-series problem in which demand has a Poisson distribution. Rosenman and Hockstra [34] investigated a problem for a repairable item in a two-level supply/repair system having Poisson distributed demands. Sherbrooke [36] analyzed a multi-product problem which consisted of repairable items in a two-level, arborescence system of stages. Simon [38] refined and extended Sherbrooke's model.

Clark [5] and Clark and Scarf [7] used dynamic programming on the single-product, stages-in-series problem with stochastic demand. In a later paper, Clark and Scarf [8] extended and refined the model. Hochstaedter [20] considered the case of a two-level, arborescence system of stages as an extension of the model. Fukuda [15], [16], extended the Clark and Scarf series model with the addition of combined ordering and disposal policies. Zacks [47], [48], formulated a Bayesian model for a two-level, parallel stages problem. Williams [45] investigated a stages-in-series problem for the backlogging case with fixed production cost.

Bessler and Veinott [4] used Veinott's dynamic process analysis technique on many forms of the stochastic, multi-stage problem. Ignall and Veinott [24] removed restrictions on initial inventory in the same formulation.

Connors and Zangwill [9] made an application of network theory to the stochastic problem, which, in principle, is an extension of the network analysis on the deterministic problem.

The greatest progress in the deterministic, nonstationary, finite horizon, multi-stage problem has come in the last ten years. In a 1966 paper, Zangwill [50] linked together single-stage models into an

acyclic network representation. The model, in effect, is a single product model. He uses the assumptions of concave production costs, piecewise concave inventory costs, known demand over a finite horizon, and backlogging to characterize a dominant set of extreme points. For the cases in which the stages are in series or in parallel networks, he develops dynamic programming algorithms to search the dominant set for optimal production schedules.

In a 1969 paper, Zangwill [49] noted that the single-product, stages-in-series case from his previous paper can be modeled as a network with conservation of flow constraints for each stage. He shows that under the concavity conditions on the costs there is an optimal schedule which is an extreme point of the network problem. Using the property that in such an extremal solution any node can have at most one incoming arc with positive flow, he presents a dynamic programming algorithm to find the optimal extreme point.

Veinott [43] shows that the single-product problem with no backlogging, as described by Zangwill, can be formulated as a Leontieff substitution model. Under the assumption of a concave objective function, the optimal solution is an extreme point. He demonstrates that when there is instant shipment of goods from one facility to another, Zangwill's network model with concave costs can be extended to include an arborescence configuration of stages with the amount of computation depending on the fourth power of the number of time periods. Under some rather severe assumptions on the cost functions, Veinott presents a simpler and more efficient procedure for the arborescence model.

Love [32] uses the network model of Zangwill for the one-product, concave cost, finite horizon, stages-in-series problem. He adds three

additional conditions: production and storage costs are separable; production costs are nonincreasing over time; and storage costs are nondecreasing over the stages. Under these conditions the optimal schedule has the property that if in a given period stage j is in production, then stage j+1 is also in production. Love uses the property to develop a more efficient dynamic programming algorithm. He also considers the stationary case where costs and demands are constant over time. It is shown that under some additional conditions a periodic optimal schedule exists, and an algorithm is given for computing it.

Crowston and Wagner [10] studied a single-product, finite horizon system which has its stages in an assembly network. Its demand is deterministic and nonstationary. Production costs are assumed to be concave, while inventory holding costs are linear. The model is an extension of the series model of Love into an assembly network, except that it does not have the concave holding costs of Love. They present two algorithms. One is a dynamic programming algorithm for the general assembly network for which solution time increases exponentially with the number of periods but only linearly with the number of stages. The other is a branch and bound algorithm and is intended for cases in which there are a large number of time periods and a nearly serial network structure.

Kalymon's [27] decomposition algorithm applies to single-product, arborescence problems which are too large for Zangwill's and Veinott's methods to solve. His model assumes holding costs are linear and that production costs, except in the latest stage on each path through the arborescence, are linear with set-up costs. The costs in the latest stages are general. The decomposition treats the latest stages as

single-stage problems and implicitly enumerates the production set-up patterns of the other stages. The number of computations increases exponentially with the number of stages having followers and linearly with the number of latest stages.

Von Lanzenauer [44] treats a multi-product problem having one machine per stage and a finite horizon. The production, holding, and shortage costs are all linear. There is also a set-up cost for each product and stage. Not every product, however, uses the machines in the same sequence. The formulation is a 0-1 program and relies on the available techniques for solution.

The multi-stage problem having an infinite horizon was examined by Taha and Skeith [41]. Their paper considers one product, the stages in series with set-up costs and linear holding costs, noninstantaneous production, delivery lags between stages, and backorders for the finished product at the final stage. They assume that the batch size at stage j is exactly enough units to build the number of units in the batch of the last stage. The optimal batch size of the last stage is found by enumeration.

Jensen and Khan [25] present a problem having one product, stages in series, and noninstantaneous production. The production rate at each stage is greater than the demand rate. The problem is to develop a start-up, shut-down schedule which equalizes average production and demand rates while minimizing set-up and inventory cost. Each stage operates in a periodic manner with a fixed cycle time over the infinite horizon. The solution technique is a dynamic program.

Crowston, Wagner, and Williams [11] treat a one-product problem whose stages have at least one incoming arc and at most one outgoing arc.

The other conditions are lot sizes which are time invariant over the infinite horizon, shipping delays which are independent of lot size, no backorders, a fixed charge per lot, and a linear holding cost. They consider only solutions which have a single lot size for each stage over time. In this situation they show that the optimal lot size for a given stage is an integer multiple of the optimal lot size of its immediate successor stage. They use dynamic programming to find the optimal lot sizes.

Schwarz and Schrage [35] use a branch and bound procedure to solve the same problem. They also present some heuristic procedures to solve the problem by optimizing pairs of adjacent stages.

Additional contributions on the problem have been made by Evans [14], Johnson and Montgomery [26], Ratliff [33], Sobel [39], Szendrovits [40], Thomas [42], and Young [46].

Zangwill, Veinott, Love, Kalymon, and Crowston and Wagner considered a multi-stage, multi-machine, single-product, deterministic, finite horizon scheduling problem. In fact, they have no constraint on the number of machines in a stage. Only Von Lanzenauer treated a multi-product, finite horizon problem having a constraint on the number of machines in a stage. The remainder of the papers which have been considered here deal with a stochastic problem or a deterministic problem with an infinite horizon. Neither of these problems falls within the category of problems studied in the following chapters.

1.3 Overview of Dissertation

The following chapters consider a multi-stage, multi-machine, multi-product scheduling problem which is deterministic and has a finite horizon. The methods used, unlike those of Von Lanzenauer, will not be

hindered by a dependence on the state of the art of 0-1 integer programming. The problem is an extension to multiple stages of the work of Dorsey, Hodgson, and Ratliff [12] on the multi-facility, multi-product problem. Like that of Dorsey et al., this problem is nonstationary and restricts the number of machines at each stage.

Two cases of the problem, distinguishable by conditions on the production rates, receive attention. In Chapter 2, the case in which a production bottleneck occurs in the initial stages is examined, and a single pass solution procedure is presented. Chapter 3 addresses the case in which the production bottleneck occurs in the final stages and shows how such a problem can be reformulated as a single-stage problem having precedences among the jobs. Due to the extreme difficulty in solving this problem, Chapter 4 presents heuristics for solution. In addition, it develops an enumeration algorithm which is used to test the accuracy and efficiency of the heuristics. Chapter 5 draws conclusions and suggests avenues for future research.

CHAPTER 2

A FRONTAL BOTTLENECK PROBLEM

2.1 Introduction

The multi-stage, multi-machine, multi-product production scheduling problem studied in this chapter is an extension of the multi-facility, multi-product problem examined by Dorsey, Hodgson, and Ratliff [12]. Where the problem of Dorsey et al. is concerned with different products manufactured in a single production operation, this problem considers different products manufactured in a series of production operations. Each operation is unique and is called a stage of production. Each product requires the same sequence of production stages in its creation as every other product. Each machine is used exclusively in a given production stage.

The gear manufacturing example in Chapter 1 illustrates these requirements. The different kinds of gears are the different products. Each kind starts from a different kind of blank. The turning of the blank and the cutting of the gear teeth are different operations and can be considered production stages. Each gear must go through both stages in the same serial sequence. Clearly, the screw machine and the gear hobber can be used only in their respective production stages.

The problem discussed in this chapter is called a frontal bottleneck problem due to an assumption that any stage of production has less production capacity than any of the succeeding stages. This assumption will cause more time per unit to be devoted to production in the earlier

of any two stages. The result is that the question of schedule feasibility becomes a question of whether or not there is enough time or machine capacity to handle the demands placed upon the earliest stages of the production system--a frontal bottleneck.

The problem discussed first in this chapter has only two stages in series. One possible solution method is presented which consists of solving a series of network flow problems. A greedy algorithm, developed by Dorsey [12], is presented which solves the network flow subproblems. It is shown that under certain assumptions the technique finds a feasible solution. With the addition of an ordering assumption on the cost coefficients, the technique finds an optimal solution. These results are then shown to extend to N stages in series. Extensions of the model and of the solution technique to more general production systems are briefly discussed. Finally, application of the technique to situations in which demand is uncertain and demand forecasts are used is considered.

2.2 Problem Definition

The problem to be considered in this chapter can be described as an industrial process in which M different products are manufactured. Each product undergoes the same two stages of production in the same sequence. Within stage j there are N_j parallel identical machines which perform the operation associated with the stage.

All production runs, called jobs, are performed on a single product and a single machine and have a duration of one time period. An (i,j) job is a production run of stage j of product i. Jobs of the same product and stage may be scheduled consecutively or at the same time on parallel, identical machines. It is assumed either that setup

is included in the production run and is performed for each job or that setups are performed between periods.

Nonnegative in-process and finished-product inventories must be maintained throughout the scheduling horizon H. A newly completed component or finished product is added to the proper inventory at the end of its production period with no time loss for transportation. The necessary raw materials are always available. The components of product i which are used as input for the production of stage 2 of product i during period t are drawn from the stage 1 in-process inventory at the end of period t-1.

Considering the gear manufacturing example, after a batch (job) of type i gear blanks has been turned, the turned blanks are sent to in-process inventory to await input to the gear-hobber stage. Each turned blank becomes one gear in finished inventory after processing by the gear hobber.

The output of stage 2 is the finished product. Demand for each kind of finished product is assumed known for each time period through the scheduling horizon. The demand for period t is satisfied from the finished-product inventory at the end of period t.

The costs of a schedule are incurred in production costs and inventory carrying costs. The production cost for a given stage and product is assumed independent of time. The inventory carrying cost for a given product is assumed a linearly increasing function of time. As a unit of a product finishes a stage of production, its inventory carrying cost per unit increases proportionally with the value added by the stage of production. The objective is to find the production schedule which minimizes the production and inventory carrying costs over the horizon H while satisfying the previously mentioned constraints.

In the next section a mathematical model is formulated for the problem. The model gives the insight that it may be possible to separate the problem and solve it as a sequence of network flow problems. Finally, a greedy algorithm, developed by Dorsey [12], is presented which solves the network flow problems.

2.3 A Mathematical Model

In order to develop some insight into the solution of the problem described in the previous section, a mathematical model will be formulated. However, it is necessary, first, to present some notation:

b_i^1	inventory carrying cost per batch (job) per period of stage 1 of product i
b_i^2	incremental inventory carrying cost (value added) per batch (job) per period of stage 2 of product i
c_i^j	production cost per batch (job) of stage j of product i
$d_{i,t}$	demand for product i in period t
D	demand matrix having entries $d_{i,t}$
H	number of periods in the scheduling horizon
I_i^j	desired level for the final inventory of stage j of product i
$I_i^j(0)$	initial inventory level of stage j of product i
M	number of kinds of finished products
N_j	number of identical machines which perform operation j
N(1,2)	minimum number of jobs in stage 1 needed to supply the input for each job in stage 2

p_i^j production rate (batch size) for stage j of product i

$w_{i,t}^j(X^{j+1})$ the number of jobs of stage j of product i which must be performed, due to the demand created by X^{j+1}, by the end of period t

$x_{i,t}^j$ number of machines which produce stage j of product i during period t

X^j matrix of decision variables for stage j (has entries $x_{i,t}^j$)

$\lceil a \rceil$ the smallest integer no less than a

$\lfloor a \rfloor$ the largest integer no greater than a

ψ_i^1 inventory carrying cost per unit per period of stage 1 of product i

ψ_i^2 incremental inventory carrying cost (value added) per unit per period of stage 2 of product i

$\delta_{t,H}$ 1, if t=H; 0, otherwise

Since a production scheduling problem is being considered, the objective is to find a schedule which satisfies the demand for finished products on time and attains the desired levels for the various final inventories. All this must be accomplished while maintaining nonnegative inventory levels, using only the available machines, and minimizing the production and inventory carrying costs.

The variable to be considered here is $x_{i,t}^j$ - the number of machines which produce stage j of product i during period t. It can also be interpreted as the number of (i,j) jobs (batches) performed during period t. The use of this variable allows the formulation of an integer program which is related to a network flow problem and, under

some limiting assumptions, lends itself to a straightforward solution technique.

Consider the following definitions. The production rate for stage j of product i is p_i^j. The initial inventory and desired level of final inventory for stage j of product i are $I_i^j(0)$ and I_i^j, respectively. The demand for product i during period t is $d_{i,t}$. Finally, the Kroenecker delta, $\delta_{t,H}$, is 1, if t=H, and 0, otherwise.

Since the finished products are the output of stage 2, the production in stage 2, aided by the initial inventory of the finished product, must satisfy the demand over the scheduling horizon and meet the desired level of final inventory of finished products. This is expressed in (2.1) for i=1, ..., M and t=1, ..., H.

$$p_i^2 \sum_{k=1}^{t} x_{i,k}^2 \geq -I_i^2(0) + \sum_{k=1}^{t} d_{i,k} + \delta_{t,H} I_i^2 \qquad (2.1)$$

Since the output of stage 1 supplies the input of stage 2, the production in stage 1, aided by the initial inventory of stage 1 components, must supply the input needs for stage 2 production and meet the desired level of final inventory of stage 1 components. If this is accomplished in every period, the in-process inventory level remains nonnegative. This stage 1 production constraint is expressed in (2.2) for i=1, ..., M and t=1,..., H. (Note: $x_{i,H+1}^j \equiv 0$.)

$$p_i^1 \sum_{k=1}^{t} x_{i,k}^1 \geq -I_i^1(0) + p_i^2 \sum_{k=1}^{t+1} x_{i,k}^2 + \delta_{t,H}\, I_i^1 \qquad (2.2)$$

Constraints (2.1) and (2.2) can be simplified and made more intuitive, if they are expressed in terms of the decision variables alone.

Solving the constraints for $\sum_{k=1}^{t} x_{i,k}^{2}$ and $\sum_{k=1}^{t} x_{i,k}^{1}$, respectively, leaves a right-hand side which is a lower bound on the number of (i,j) jobs which must be completed by the end of period t. (The production in stage 2 causes a demand on stage 1.) Let D and X^2 be the matrices which have the entries $d_{i,t}$ and $x_{i,t}^{2}$, respectively. Also, let $\lceil a \rceil$ be the smallest integer no less than a. Then, the maximum number of (i,j) jobs which must be completed by the end of period t due to the demand in the argument is expressed by $w_{i,t}^{j}(X)$, defined by

$$w_{i,t}^{2}(D) \equiv \max \left\{0, \left\lceil (-I_i^2(0) + \sum_{k=1}^{t} d_{i,k} + \delta_{t,H}\, I_i^2)/p_i^2 \right\rceil \right\}$$

and

$$w_{i,t}^{1}(X^2) \equiv \max \left\{0, \left\lceil (-I_i^1(0) + p_i^2 \sum_{k=1}^{t+1} x_{i,k}^{2} + \delta_{t,H}\, I_i^1)/p_i^1 \right\rceil \right\}$$

Using the w functions, constraints (2.1) and (2.2) will become (2.9) and (2.8), respectively, in the complete problem formulation (P2.1), below.

A straightforward stage 2 constraint requires that production in stage 2 during period 1 does not exceed the input available from the stage 1 initial inventory for all products i=1, ..., M.

$$p_i^2 x_{i,1}^2 \le I_i^1(0) \tag{2.3}$$

Solving for $x_{i,1}^2$ and letting $\lfloor a \rfloor$ be the largest integer no greater than a, (2.3) can be transformed (since $x_{i,1}^2$ is integer) into a tighter constraint ((2.10) in the complete problem formulation (P2.1), below).

Let N_j be the number of machines available in stage j. Constraint (2.4) ensures for j=1, 2 and t=1, ..., H that no more than N_j machines are used.

$$\sum_{i=1}^{M} x_{i,t}^{j} \leq N_j \tag{2.4}$$

Let c_i^j be the production cost per (i,j) job. Also, let ψ_i^j be the inventory carrying cost (incremental for stage 2) per (i,j) component per period. The objective function is to minimize total cost and can be expressed as in (2.5).

$$\min \sum_{i=1}^{M} \sum_{j=1}^{2} \left[c_i^j \sum_{t=1}^{H} x_{i,t}^j + \psi_i^j p_i^j \sum_{t=1}^{H} (H-t+1) x_{i,t}^j \right] \tag{2.5}$$

Since total cost is being minimized, the minimum necessary number of jobs will be performed. Thus, in an optimal solution,

$$\sum_{t=1}^{H} x_{i,t}^2 = w_{i,H}^2(D) \qquad \text{and}$$

$$\sum_{t=1}^{H} x_{i,t}^1 = w_{i,H}^1(X^2), \qquad \text{both constants.}$$

Consequently, with a little manipulation, (2.5) can be restated as

$$\min \sum_{i=1}^{M} \sum_{j=1}^{2} \sum_{t=1}^{H} -t\Psi_i^j p_i^j x_{i,t}^j + \text{constant} . \tag{2.6}$$

The production costs have dropped out of the optimization, which leaves only the inventory costs. Let the notation be consolidated by defining $b_i^j = \psi_i^j p_i^j$.

The objective function (2.7) in P2.1 is derived from (2.6) and has the effect of minimizing inventory cost. Note that the objective tends to schedule all the jobs as late as possible. This objective function makes sense since the later a job is scheduled, the less time its output will be held in inventory.

The mathematical model can be expressed as the following integer program:

$$\max \sum_{i=1}^{M} \sum_{j=1}^{2} \sum_{t=1}^{H} tb_i^j x_{i,t}^j \;; \tag{2.7}$$

s.t.

$$\sum_{i=1}^{M} x_{i,t}^j \leq N_j \,, \qquad \text{all } j \text{ and } t; \tag{2.4}$$

P2.1

$$\sum_{k=1}^{t} x_{i,k}^1 \geq w_{i,t}^1(X^2) \,, \qquad \text{all } i \text{ and } t; \tag{2.8}$$

$$\sum_{k=1}^{t} x_{i,k}^2 \geq w_{i,t}^2(D), \qquad \text{all } i \text{ and } t; \tag{2.9}$$

$$x_{i,1}^2 \leq \left\lfloor I_i^1(0)/p_i^2 \right\rfloor, \qquad \text{all } i; \tag{2.10}$$

$$x_{i,t}^j \geq 0 \text{ and integer.} \tag{2.11}$$

For j=2 (thus, considering only the second production stage), (2.7), (2.4), (2.9), (2.10), and (2.11) define the problem (with a unimodular constraint matrix) addressed by Dorsey, Hodgson, and Ratliff [12] . Assuming stage 1 can supply the input required by stage 2, the stage 2 problem can be solved using a network flow algorithm. The decision variables from the stage 2 solution define X^2. Considering only j=1 and fixing X^2, the remainder of P2.1 ((2.7), (2.4), (2.8), and (2.11)) defines the stage 1 problem. The stage 1 problem also has a unimodular constraint matrix and can be solved by a network flow algorithm.

An example is presented in Figure 2.1. The figure is the constraint matrix of P2.1 for two products and three periods, assuming

$x^1_{1,1}$	$x^1_{1,2}$	$x^1_{1,3}$	$x^1_{2,1}$	$x^1_{2,2}$	$x^1_{2,3}$	$x^2_{1,1}$	$x^2_{1,2}$	$x^2_{1,3}$	$x^2_{2,1}$	$x^2_{2,2}$	$x^2_{2,3}$
1	0	0	1	0	0	0	0	0	0	0	0
0	1	0	0	1	0	0	0	0	0	0	0
0	0	1	0	0	1	0	0	0	0	0	0
0	0	0	0	0	0	1	0	0	1	0	0
0	0	0	0	0	0	0	1	0	0	1	0
0	0	0	0	0	0	0	0	1	0	0	1
1	0	0	0	0	0	0	0	0	0	0	0
1	1	0	0	0	0	0	0	0	0	0	0
1	1	1	0	0	0	0	0	0	0	0	0
0	0	0	1	0	0	0	0	0	0	0	0
0	0	0	1	1	0	0	0	0	0	0	0
0	0	0	1	1	1	0	0	0	0	0	0
0	0	0	0	0	0	1	0	0	0	0	0
0	0	0	0	0	0	1	1	0	0	0	0
0	0	0	0	0	0	1	1	1	0	0	0
0	0	0	0	0	0	0	0	0	1	0	0
0	0	0	0	0	0	0	0	0	1	1	0
0	0	0	0	0	0	0	0	0	1	1	1
0	0	0	0	0	0	1	0	0	0	0	0
0	0	0	0	0	0	0	0	0	1	0	0

Figure 2.1. Constraint matrix.

the righthand side of (2.8) is fixed. The first six rows are (2.4), the second and third sets of six rows are (2.8) and (2.9), respectively. The final two rows are (2.10). Linear transformations on rows seven through eighteen can be used to create a network flow constraint matrix. The second stage can be solved to determine the righthand side of (2.8). Then stage 1 can be solved.

Clearly, then as a heuristic approach, it appears that problem P2.1 can be separated into two subproblems which, if solved in the proper order, may yield a solution to the full problem. In the next section an ordering procedure and a greedy algorithm (Dorsey [12]) for solving the individual stage subproblems is presented.

2.4 The Backward Solution Technique

In the last section it was shown that (as a heuristic) problem P2.1 can be separated into two parts, each of which can be solved as a network flow problem (if the part corresponding to stage 2 is solved first). The problem of stage 2 is the multi-facility, multi-product production scheduling problem solved by Dorsey, Hodgson, and Ratliff [12]. After solution of stage 2, stage 1 has the same form. Dorsey developed a greedy algorithm which is much more efficient than standard network flow techniques.

The following procedure, called the Backward Solution Technique or BST, is applied to two-stage production problems.

Step 1. Set j=2 and use the demand for the stage j problem.

Step 2. Solve the network flow problem presented by stage j, using Dorsey's algorithm [12]. Use the result to define X^j, the schedule for stage j.

Step 3. If $j > 1$, set $j=j-1$, use X^{j+1} as the demand for the stage j problem, and go to Step 2. Otherwise, stop.

An application of the BST to the gear manufacturing problem would cause the teeth-cutting stage to be optimally scheduled first. This schedule would act as a demand timetable for turned gear blanks to be used as input for the gear hobbers. The gear-blank-turning stage is then solved optimally using the demand from the gear hobbers.

It remains to establish conditions for which the BST gives an optimal, or even a feasible, solution to the scheduling problem P2.1. First, however, Dorsey's algorithm and its use in the BST need to be explained. Dorsey's algorithm is presented primarily because much of the later development in this chapter depends on its structure. (Note that the following discussion considers a single-stage problem.)

In order to present Dorsey's algorithm, the concept of a relative deadline for scheduling a job must be explained first. The relative deadline of a job is that period in the scheduling horizon in which the first unit of the output of the job is used to satisfy demand. This period can be determined before scheduling takes place. Thus, a job may not be scheduled later than its relative deadline.

In order to determine relative deadlines, it is necessary to be able to distinguish between two jobs of the same product. In some (not necessarily optimal) schedule for a set of identical jobs, number the jobs according to their relative positions in the schedule. If two jobs are in the same period, then the job on the higher-numbered machine has the higher number. Thus, the $n+1^{st}$ job of a product is "later" than the n^{th} job. Without loss of generality the n^{th} job of the product in this schedule is the n^{th} job of the product in any optimal

schedule. Also, a first-in-first-out inventory system is assumed for ease of notation and without loss of generality.

Using numbered jobs and a FIFO inventory system, initial inventory satisfies the early demand. After that the first unit of demand is satisfied by the first unit of output from the first job. The period in which this happens is the relative deadline of the first job, etc. In making these calculations the desired level of the final inventory is considered part of the demand in the last period.

In his scheduling procedure Dorsey [12] considers the periods one at a time, starting with the last period. Within the period, he starts by scheduling the jobs of the highest-numbered product. He schedules as many as possible of these jobs which have not already been scheduled and which have relative deadlines no earlier than the period under consideration. Having scheduled one product, he starts with the next lower-numbered product. Having completely scheduled a period, he considers the next earlier period.

Dorsey [12] has shown that the method finds an optimal schedule, if the products are numbered according to their nondecreasing costs.

Consider Table 2.1A for a two-machine example of the use of Dorsey's method. The entries of the table are the demand (in units) for the two products in the example. Products 1 and 2 have production rates of two and three units per period, respectively. Product 2 has an initial inventory of four units. Since period 4 has a demand for three units of product 1, it will take two jobs to produce them. So the first two product 1 jobs have their relative deadline at period 4. This leaves one unit in product 1 inventory going into period 5.

TABLE 2.1A. Single-stage example problem-demand table.

TIME	1	2	3	4	5	6	7
PRODUCT 1	0	0	0	3	2	1	2
PRODUCT 2	0	0	0	8	4	4	3

TABLE 2.1B. Relative deadlines for the problem in Table 2.1A.

TIME	1	2	3	4	5	6	7
PRODUCT 1	0	0	0	2	1	0	1
PRODUCT 2	0	0	0	2	1	1	1

However, that is not enough to cover the demand in period 5--another product 1 job is needed and has its relative deadline in period 5. Because of its initial inventory only two jobs are needed to satisfy demand for product 2 in period 4. Table 2.1B has as its entries the number of jobs of each product which have their relative deadlines in the indicated periods. Table 2.1B shows that periods 5 through 7 have enough machines so that their jobs can be scheduled at their deadlines. However, period 4 has deadlines for four jobs but room for only two. The scheduling method places the higher-numbered products in period 4 and considers the others in period 3. Figure 2.2 is a Gantt chart, the entries of which are the product numbers of the jobs scheduled in the indicated periods.

Consider again a two-stage problem. In order to use Dorsey's method in a stage other than the last stage of production, it is necessary to be able to use the concept of the relative deadline in an earlier stage. Demand is taken from inventory at the end of the period in which it appears; however, input to a later stage is taken from inventory at the end of the period immediately before it is needed. As a convention to rectify this problem let the input requirement for stage 2 at time t be regarded as demand on stage 1 at time t-1. The relative deadline for jobs in stage 1 depends on the demand placed on stage 1 by the schedule calculated for stage 2. Dorsey's method can now be applied to stage 1 in a straightforward manner.

Let us use the example in Tables 2.1A and 2.1B and in Figure 2.2 as the second stage of a two-stage problem. Thus, $p_1^2=2$, $p_2^2=3$, $I_2^2(0)=4$,

TIME	1	2	3	4	5	6	7
MACHINE 1			1	2	1		1
MACHINE 2			1	2	2	2	2

FIGURE 2.2. Dorsey's schedule for the problem in Table 2.1A.

and $I_1^2=1$. Let stage 1 have two machines, $p_1^1=p_2^1=2$, and $I_2^1=1$. The stage 1 "demand" derived from the stage 2 schedule in Figure 2.2 is in Table 2.2A. The demand translates into the stage 1 relative deadlines in Table 2.2B. From this, Dorsey's method derives the stage 1 schedule in the Gantt chart of Figure 2.3.

The stage 2 and stage 1 schedules in Figures 2.2 and 2.3, respectively, represent the BST solution to the two-stage problem which uses the demands in Table 2.1A.

The Backward Solution Technique and the method used to solve the individual stages within the BST were presented in this section. The next section presents the conditions under which the BST finds feasible and optimal solutions to the two-stage problem.

2.5 Feasibility and Optimality Conditions for the Backward Solution Technique

The purpose of this section is to develop conditions which enable the BST to find feasible and (under additional conditions) optimal solutions to problem P2.1. Three assumptions will be presented. It will be proven that under the three assumptions, the BST will find a feasible solution, if one exists. A fourth assumption will be made. It will be shown that under the four assumptions the BST will find an optimal solution, if one exists.

Consider the Gantt chart in Figure 2.4. The chart is the combination of Figures 2.2 and 2.3, but depicts only the schedule of the product 2 jobs in the two-stage problem. (The first row for each stage is machine 1.) Since $p_2^1 = 2$ and $p_2^2 = 3$, the jobs in period 2 and 3 of stage 1 furnish the input for the two jobs in period 4 of stage 2.

TABLE 2.2A. First stage demand caused by the stage 2 schedule in Figure 2.2.

TIME	1	2	3	4	5	6	7
PRODUCT 1	0	4	0	2	0	2	0
PRODUCT 2	0	0	6	3	3	3	1

TABLE 2.2B. Relative deadlines for the stage 1 problem in Table 2.2A.

TIME	1	2	3	4	5	6	7
PRODUCT 1	0	2	0	1	0	1	0
PRODUCT 2	0	0	3	2	1	2	0

TIME	1	2	3	4	5	6	7
MACHINE 1	1	1	2	2	1	2	
MACHINE 2	1	2	2	2	2	2	

FIGURE 2.3. Dorsey's schedule for the stage 1 problem in Table 2.2A.

TIME	1	2	3	4	5	6	7
STAGE 1			2	2		2	
		2	2	2	2	2	
STAGE 2				2			
				2	2	2	2

FIGURE 2.4. Two-stage schedule of product 2.

All of the output of the first and third jobs of stage 1 (machine 2 of periods 2 and 3, respectively) are used as input strictly to the first and second jobs, respectively, of stage 2. The second job of stage 1 supplies its first unit of output to the first job of stage 2 and its second unit to the second job of stage 2. Therefore, the relative deadlines of the first and second jobs of stage 1 are determined by (in fact, are one period earlier than) the period in which the first job of stage 2 is scheduled. If the first job in stage 2 moved into period 3, the second job of stage 1, in order to supply the job, would be forced to move into period 2. If that stage 2 job moved to period 2, it would force the first two stage 1 jobs into period 1.

The first two stage 1 jobs in Figure 2.4 are supplier jobs of the first stage 2 job. A job of stage 1 is a <u>supplier job</u> of a stage 2 job if the first unit of output from the stage 1 job is used as input to the stage 2 job. Also, a job of stage 2 is a <u>consumer job</u> of a stage 1 job if the stage 2 job uses as input the first unit of output from the stage 1 job. Consequently, the first stage 2 job in Figure 2.4 is the consumer job of the first two stage 1 jobs, and the second stage 2 job is the consumer job of the third stage 1 job.

The first assumption to be used in the proof that the BST finds feasible solutions concerns the relative production rates (batch sizes) of the same product between the two stages.

<u>Assumption 2.1</u>: (production rate) The production rate $p_i^1 \leq p_i^2$ for all i.

The consequence of the production rate assumption is that each (i,2) job requires an input at least as great as the output of one (i,1) job. In fact, each (i,2) job requires input equivalent to the output of p_i^2/p_i^1 (i,1) jobs.

As an example of the possible consequences of violation of Assumption 2.1, consider the Gantt chart in Figure 2.5A. In this problem each stage has one machine, $2p_1^1 = p_1^2$, $p_2^1 = 2p_2^2$, and all stage 2 jobs have their relative deadlines at period 4. There is no initial or desired final inventory. Scheduling a job in period 0 is infeasible. The BST solution to the problem is in Figure 2.5A. It is infeasible; however, the feasible solution is in Figure 2.5B.

<u>Assumption 2.2</u>: (supplier job) Each stage 2 job has at least N(1,2) stage 1 supplier jobs, where $N(1,2) = \min_i \left\lfloor p_i^2/p_i^1 \right\rfloor$.

When Assumption 2.1 is considered, $N(1,2) \geq 1$, and Assumption 2.2 states that each stage 2 job has at least one supplier job in stage 1.

In the early periods of the scheduling horizon, demand can cause the first stage 2 job of a product to be scheduled so early that there is not enough time to schedule its supplier jobs. Another possibility is that initial inventory of stage 1 of product i is so high that the first (i,2) job does not need supplier jobs for its input. The supplier job assumption avoids these possibilities by forcing each stage 2 job to have at least N(1,2) supplier jobs. Note that this assumption effectively places an upper bound on initial in-process inventories. Practical considerations of this upper bound will be discussed in a later section. Furthermore, Assumption 2.2 ensures that if a product is produced in stage 2, it has a component produced in stage 1, since it must have a supplier job.

TIME	0	1	2	3	4
STAGE 1	1	1	2		
STAGE 2			1	2	2

FIGURE 2.5A. Production rate counterexample-BST solution.

TIME	0	1	2	3	4
STAGE 1		2	1	1	
STAGE 2			2	2	1

FIGURE 2.5B. Production rate counterexample-feasible solution.

Figure 2.6A presents an example of the violation of the supplier job assumption. Each stage has one machine, $p_1^1 = p_1^2$, $p_2^1 = p_2^2$, $I_2^1(0) = p_2^2$, and all stage 2 jobs have their relative deadlines at period 3. Since $I_2^1(0) = p_2^2$, the first (2,2) job has no supplier job. The BST solution is in Figure 2.6A, and the feasible solution is in Figure 2.6B.

<u>Assumption 2.3</u>: (machine availability) The number of stage 1 machines satisfies $N_1 \leq N_2 N(1,2)$.

Without the truncation in N(1,2), Assumptions 2.2 and 2.3 state $N_1 \leq N_2 \min_i (p_i^2/p_i^1)$ or $p_i^1 N_1 \leq p_i^2 N_2$ for all i. Thus, the one-period production capability of stage 1 is no greater than the one-period production capability of stage 2 for any product and the production bottleneck is maintained in stage 1. The consequence of the machine availability assumption, when combined with Assumptions 2.1 and 2.2, is that there are no more machines available in any period t of stage 1 than are necessary to execute the minimum number of supplier jobs which N_2 jobs in period t+1 of stage 2 can have. In other words, if there are no idle machines in period t+1 of stage 2 and all stage 1 jobs are scheduled as late as possible, there are no idle machines in period t of stage 1.

In the example of Figure 2.7A stage 1 has too many machines, in violation of Assumption 2.3. Stage 1 has two machines, stage 2 has one machine, and all stage 2 jobs have their relative deadlines at period 3. The production rates $3p_1^1 = p_1^2$ and $p_2^1 = p_2^2$; thus, N(1,2) = 1. Figure 2.7A presents the BST solution, and Figure 2.7B presents the feasible solution.

TIME	0	1	2	3
STAGE 1	1		2	
STAGE 2		1	2	2

FIGURE 2.6A. Supplier job counterexample-BST solution.

TIME	0	1	2	3
STAGE 1		1	2	
STAGE 2		2	1	2

FIGURE 2.6B. Supplier job counterexample-feasible solution.

TIME	0	1	2	3
STAGE 1	1	1		
		1	2	
STAGE 2			1	2

FIGURE 2.7A. Machine availability counterexample-BST solution.

TIME	0	1	2	3
STAGE 1		1	1	
		2	1	
STAGE 2			2	1

FIGURE 2.7B. Machine availability counterexample-feasible solution.

It has been shown that violation of any of the Assumptions 2.1, 2.2, or 2.3 can result in the BST failing to find a feasible solution, when one exists. It will be shown that when all three assumptions are met, the BST finds a feasible solution, if one exists. First, however, some lemmas must be proven.

The first lemma characterizes a solution to a single-stage problem obtained by Dorsey's method. It concerns the ordering of the jobs by product number within a schedule.

Lemma 2.1: Consider a schedule, found by Dorsey's algorithm, to a single-stage problem. If a job of some product i is scheduled in period t1 earlier than its relative deadline t2, then every machine in the interval [t1 + 1, t2] is utilized by the schedule, and every job in the interval has a product number which is at least as high as i.

Proof: Assume for the purpose of contradiction that a machine in some period t in the interval [t1 + 1, t2] is idle or scheduled to process a job of product i1, where i1 < i. When Dorsey's algorithm scheduled period t, it assigned to t all product i jobs which were unscheduled and had relative deadlines no earlier than t. This assignment was made before any jobs of product i1 were assigned to t and before any decision was made to leave a machine in t idle. Consequently, no job of product i which has a relative deadline in period t or later is scheduled earlier than period t. This contradicts the existence of the product i job referred to in the hypothesis of the lemma. Therefore, every machine in period t is utilized, and every job in period t has a product number which is at least as high as i, for all t in [t1 + 1, t2]. Q.E.D.

The insight to be found in Lemma 2.1 is that in a Gantt chart of Dorsey's solution all the machine blocks between a given job and its relative deadline contain jobs having product numbers no lower than the product number of the job under consideration.

Lemma 2.2: If there exists a feasible solution to a single-stage problem, then Dorsey's algorithm will find a unique, feasible solution.

Proof: Follows from Dorsey [12] . Q.E.D.

The preceding assumptions and lemmas are used in Lemma 2.3, which shows that, under certain conditions, a pairwise interchange of two jobs in the second stage of a feasible schedule for a two-stage problem can be made so that the resulting schedule is feasible.

In considering Lemma 2.3, refer to the Gantt chart in Figure 2.8. Unlike previous Gantt charts in this chapter, each stage has an unspecified number of machines, except that they satisfy the machine availability assumption. The i3 in some of the periods represents a class of products (not necessarily all of which have the same product number) all of which have higher product numbers than i1 and i2. The i4 in other periods represents another class of products. The i1 and i2 in the chart indicate that each of those periods contains at least one job of product i1 or i2, among other jobs.

Note in the proof of the lemma that all changes in the solution are accomplished by applying Dorsey's algorithm to a stage or by a pairwise interchange of jobs in which the job of the higher-numbered product moves to a later period and the job of the lower-numbered product moves to an earlier period.

TIME	t1	...	t2	...	t3-1	t3	...	t4-1	t4	...	t5
STAGE 1	i2	...	i4	...	i4	i4	...	i1		...	
STAGE 2		...	i2	...	i3	i3	...	i3	i1	...	

FIGURE 2.8. Portions of a solution for two adjacent stages.

Lemma 2.3: Consider the Gantt chart of a feasible solution to a two-stage problem in which the first stage has a feasible solution found by Dorsey's algorithm and in which each stage 2 job is scheduled so that there are no idle machines between it and its relative deadline. For any two products i1 and i2 such that i1 < i2, consider an (i1,2) job in some period t4 and an (i2,2) job in some period t2 (t2 < t4) both of which have relative deadlines no earlier than t4. Furthermore, let i1 be the lowest product number in period t4 of stage 2 and let this (i1,2) job be the "earliest" (lowest job number) of the (i1,2) jobs in period t4. Finally, let the (i2,2) job be the "latest" (highest job number) of the (i2,2) jobs in period t2. If

(a) the jobs in the interval [t2+1, t4-1] of stage 2 are of the product class i3,

(b) the (i1,2) job and the (i2,2) job are interchanged in the schedule, and

(c) Assumptions 2.1, 2.2, and 2.3 are met,

then there exists a feasible solution which is identical to the new solution in stage 2 and has a feasible solution from Dorsey's algorithm in stage 1.

Proof: It will be shown that all the (i1,1) supplier jobs of the (i1,2) job which are in the interval [t2, t4-1] are actually in period t4-1. It will also be shown that there are no more than N(1,2) of these (i1,1) supplier jobs in period t4-1. Thus, when the interchange of their (i1,2) consumer job and the (i2,2) job is made, the N(1,2) (i1,1) supplier jobs in t4-1 can interchange feasibly with N(1,2) (i2,1)

supplier jobs of the (i2,2) job in periods earlier than t2. The resulting solution is feasible. Finally, Dorsey's algorithm is applied to the new stage 1 solution.

Consider Figure 2.8. Let period t1(<t2) be the latest period containing (i2,1) supplier jobs of the (i2,2) job in period t2. Define all the stage 1 jobs in the interval [t1+1, t4-2] to be in the product class i4. The (i2,1) supplier job in period t1 has its relative deadline in period t2-1. By Lemma 2.1, each job in the interval [t1+1, t2-1] has a product number no lower than i2, therefore, higher than i1. By Assumptions 2.1 and 2.2, each stage 2 job in the interval [t2+1, t4-1] has at least N(1,2) supplier jobs. Then, by Assumption 2.3 and Lemma 2.1, all stage 1 jobs in the interval [t2, t4-2] have product numbers at least as high as those of the products in class i3, which are higher than i1 and i2. Therefore, all products in the class i4 have numbers at least as high as i2 and higher than i1. Consequently, there are no (i1,1) jobs in the interval [t1+1, t4-2].

Since the (i1,2) job of interest in period t4 is the lowest-numbered job of the lowest-numbered product in period t4 and, by hypothesis, there are no idle machines between the (i2,2) job in period t2 and its relative deadline, then period t4 of stage 2 has N_2 jobs, N_2-1 of which have higher product numbers or higher job numbers than the (i1,2) job under consideration. By all three assumptions and Lemma 2.1, there are at least (N_2-1)N(1,2) supplier jobs which have higher product or job numbers than and are scheduled later than the supplier jobs of the (i1,2) job (N(1,2) for each of the other jobs in period t4 of stage 2.) Therefore, period t4-1 contains <u>at most</u>

N(1,2) supplier jobs of the "earliest" (lowest-numbered) (i1,2) job in period t4.

Interchange the highest-numbered (i2,2) job in period t2 and the lowest-numbered (i1,2) job in period t4. (The interchange is feasible, since each job has its relative deadline at least as late as period t4.)

It has been shown that the (i1,2) job in the interchange has <u>at most</u> N(1,2) supplier jobs in the interval [t1+1, t4-1] (all in period t4-1). By the supplier job assumption, the (i2,2) job in the interchange has <u>at least</u> N(1,2) supplier jobs in period t1 or earlier. In order to maintain feasibility in stage 1 in conjunction with the stage 2 interchange, the period t4-1 supplier jobs of the (i1,2) job in the stage 2 interchange must be interchanged with the "latest" supplier jobs of the (i2,2) job in the stage 2 interchange.

These stage 1 interchanges must preserve the ordering of the jobs of the same product. There are two cases to be considered.

Case 1: There are <u>no</u> (i2,1) jobs in the interval [t1, t4-2] which are "later" (higher-numbered) than the supplier jobs of the (i2,2) job in the interchange.

Case 2: There are (i2,1) jobs in the interval [t1, t4-2] which are "later" (higher-numbered) than the supplier jobs of the (i2,2) job in the interchange.

If Case 1 is true, moving the (i2,1) supplier jobs to period t4-1 does not alter the ordering of the (i2,1) jobs. By Lemma 2.1, no (i1,1) jobs lie between the supplier jobs of the (i2,2) job in the stage 2 interchange and the supplier job's former relative deadline, period t2. Consequently, moving the (i1,1) supplier jobs to the

locations of the (i2,1) supplier jobs does not alter the ordering of the (i1,1) jobs. Therefore, make the necessary stage 1 pairwise interchanges.

If case 2 is true, it is shown in Lemma 2.1 that there are no (i1,1) jobs in period t4-1. No stage 1 interchanges are necessary.

After the stage 2 interchange and the stage 1 interchanges (if any are necessary) are made, both stages 1 and 2 have feasible schedules. The only thing which remains to be done is to convert the stage 1 feasible solution to a feasible solution based on Dorsey's algorithm. By Lemma 2.2, applying Dorsey's algorithm to the stage 1 problem created by the new stage 2 solution completes the proof.

Q.E.D.

The results of Lemma 2.3 form the cornerstone of the proof of the following theorem:

Theorem 2.1: If there exists a feasible solution to the two-stage problem and if Assumptions 2.1, 2.2, and 2.3 are met, the Backward Solution Technique finds a feasible solution.

Proof: The proof consists of assuming the existence of a feasible solution to a two-stage problem and then converting it to a feasible solution both stages of which are Dorsey solutions. After transforming stage 1 into a Dorsey solution, the stage 2 feasible solution is compared to a Dorsey solution of stage 2. Starting with the last period, the stage 2 solution is converted by pairwise interchanges to the Dorsey solution, period by period. After each interchange the stage 1 Dorsey solution is updated to maintain feasibility.

Consider any feasible solution to the two-stage problem. By Lemma 2.2, stage 1 of the solution can be converted to a Dorsey solution. The solution now is feasible in stage 2 and has a Dorsey solution in stage 1. Order the jobs on the machines in each period of stage 2 by increasing product and job number. If there are any idle machines in stage 2 between a job and its relative deadline, move the job to the latest such machine.

Also consider the Dorsey solution to stage 2. Compare the stage 2 feasible and Dorsey solutions. (The conditions of Figure 2.8 and Lemma 2.3 will now be created.) Find the latest period in which the two stage 2 Gantt charts don't agree. Call that period t5. In period t5 of the stage 2 Dorsey schedule find the highest-numbered job of the highest-numbered product which is <u>not</u> in period t5 of the stage 2 feasible solution. Call that job n2 of product i2. Find the period in the stage 2 feasible solution which contains job n2 of product i2. Call that period t2. Since the two stage 2 schedules agree in the interval [t5+1, H], t2 < t5. By the ordering of job numbers and the method of choice of job n2, job n2+1 of product i2 is in the interval [t5, H] in both stage 2 schedules. Thus, job n2 of product i2 is the highest-numbered (i2,2) job in period t2 of the feasible schedule. Job n2 of product i2 must now work its way, through a series of pairwise interchanges, to period t5 of stage 2 of the feasible solution. (Until it reaches t5, all discussion concerns the feasible solution only.)

The next step is to find a job with which to interchange job n2 of product i2. Find the earliest period which is later than t2 and which contains a product number smaller than i2. Call the period t4

and call the product number i1 (t2 < t4 and i1 < i2). Let job n1 be the lowest-numbered (i1,2) job in period t4. Let i3 be the class of products in the stage 2 interval [t2+1, t4-1]. By the choice of t4 and job number ordering, all the products in the class i3 have higher numbers than i2. Interchange job n2 of product i2 and job n1 of product i1. Find the new feasible Dorsey schedule for stage 1, the existence of which is shown in Lemma 2.3.

If t4 < t5, job n2 is the only (i2,2) job in period t4 (job n2+1 is in period t5 or later). Redefine t2 to be the old t4. Search for a new period t4, product i1, and job n1.

When job n2 reaches period t5 (t4 = t5), find a new period t5, product i2, and job n2. The search must eventually end with the conversion of period 1 into period 1 of the stage 2 Dorsey solution, because the interval [t5,H] has no schedule alterations (there is no cycling in the search).

The feasible solution now is a Dorsey solution in both stages. Lemma 2.2 shows that Dorsey's algorithm finds unique solutions. Since the BST finds Dorsey solutions at each stage, this two-stage solution is the one the BST would find.

Q.E.D.

Since no mention was made of an objective function in the proof, there is the following corollary:

Corollary 2.1: If there is a feasible solution to the two-stage problem and if Assumptions 2.1, 2.2, and 2.3 are met, then the BST finds a feasible solution for any objective function.

It remains to be shown under what additional conditions the BST will find an optimal solution. The fourth assumption requires that the products can be ordered equivalently in each stage by their cost b_i^j.

Assumption 2.4: (cost) The inventory cost function $b_i^j \le b_{i+1}^j$ for

$$i = 1, \ldots, M-1 \text{ and } j = 1, 2 .$$

The meaning of the cost assumption in the first stage is that the inventory carrying cost of a batch of product i is no greater than the inventory carrying cost of a batch of product i+1. In stage 2 the cost is the value added to a batch, but the relationship between products must still hold.

The assumption of the relationship between costs, when considered for an individual stage, is the same assumption Dorsey [12] makes to ensure that his method finds an optimal solution. Thus, under the cost assumption, the BST finds a solution which is optimal in each stage, if a solution exists for each stage. However, that does not mean the total solution is optimal.

To show that the BST may not find an optimal solution if Assumption 2.4 is not met, consider a two-stage problem. Each stage has one machine, each stage 2 job has its relative deadline at the end of period 3, $p_1^1 = p_1^2$, and $p_2^1 = p_2^2$. Let $b_1^1 = 3$, $b_2^1 = 1$, $b_1^2 = 1$, and $b_2^2 = 2$. Figure 2.9A contains the BST solution, which has an objective value of 13. The optimal solution, in Figure 2.9B, has an objective value of 14. (Remember, that the objective is to maximize.)

TIME	1	2	3
STAGE 1	1	2	
STAGE 2		1	2

FIGURE 2.9A. Cost counterexample-BST solution.

TIME	1	2	3
STAGE 1	2	1	
STAGE 2		2	1

FIGURE 2.9B. Cost counterexample-optimal solution.

Theorem 2.2: If the Assumptions 2.1, 2.2, 2.3, and 2.4 are met and if a feasible solution exists to the two-stage problem, then the Backward Solution Technique finds an optimal solution.

Proof: Consider any optimal solution. Clearly, since it is optimal, no idle machines lie between a job and its relative deadline. Convert the optimal solution to a solution which has Dorsey solutions in each stage by the same conversion method as is used in the proof of Theorem 2.1. There are four ways in which the solution is changed. Reordering the jobs within a stage does not change their objective value. Since there are no idle machines between a job and its relative deadline, the jobs of stage 2 will not be moved later to fill idle machines. In the proofs of Lemma 2.3 and Theorem 2.1 all pairwise interchanges moved a job of a higher-numbered product (i2) later and a job of a lower-numbered product (i1) earlier, for a net objective value change per period moved of $b^j_{i2} - b^j_{i1} \geq 0$ (by Assumption 2.4). Finally, stage 1 is periodically solved using Dorsey's algorithm. Dorsey [12] showed that, under Assumption 2.4 applied to a single stage, his algorithm finds an optimal solution. Thus, under Assumption 2.4, applying Dorsey's method to a solution in stage 1 does not worsen the solution.

It has been shown that each alteration which must be made to an optimal solution to a two-stage problem in order to convert it to a solution having Dorsey solutions in each stage results in a solution at least as good as the optimal solution. Consequently, under Assumptions 2.1, 2.2, 2.3, and 2.4, a solution which has a Dorsey solution in both of its stages is optimal (if a feasible solution exists).

Since the BST generates unique Dorsey solutions in each stage and finds a feasible solution (Lemma 2.2 and Theorem 2.1), it finds that optimal solution.

Q.E.D.

It has been shown that under the production rate, supplier job, and machine availability assumptions the BST finds a feasible solution to the two-stage problem (if a feasible solution exists), regardless of objective function. With the addition of the cost assumption, the BST finds the optimal solution. The combination of the production rate and machine availability assumptions causes the frontal bottleneck discussed earlier. The posterior bottleneck for which $p_i^1 \geq p_i^2$ for all i and for which there is a lower bound on the number of machines in stage 1 is discussed in the next chapter. The cases in which $p_i^1 \leq p_i^2$ for some i and $p_i^1 \geq p_i^2$ for other i are very difficult to analyze and are not discussed here. If $p_i^1 > p_i^2$, there are some product i jobs in stage 2 which do not require supplier jobs (the supplier job assumption would not hold). All the other assumptions are independent of each other. The supplier job assumption affects the allowable level of initial inventory and is very restrictive. It is shown, however, in the next section that violation of the supplier job assumption results in start-up problems, which are resolved after a short interval. The cost assumption appears to be reasonable in that an expensive (or valuable) product tends to remain expensive relative to the other products as it goes through the stages of production. In the next section, practical applications of the BST are considered, such as problems in which demand may fluctuate and/or initial inventory levels may cause the violation of the supplier job assumption.

2.6 Application of the Backward Solution Technique

The normal use of a scheduling technique like the BST is to schedule production over a horizon using all available information on demand and inventory. After performing one period's production, the schedule is then recomputed using updated demand and inventory information. This cycle is repeated for the remainder of the production process.

In computing the schedules it becomes apparent that initial, in-process inventory may cause Assumption 2.2, that each (i,j) job has at least N(j,j+1)(i,j) supplier jobs, not to be realized for early jobs. In a practical sense, this situation is a start-up problem, however, and after a period of time resolves itself. Let H_1 be the period in the first schedule by which each product has had completed at least one stage 2 job having a supplier job. From no later than H_1 on, Assumption 2.2 is satisfied in the first schedule, which must be optimal for all stage 2 jobs scheduled after H_1 and for their suppliers. Thus, the start-up period ends no later than period H_1.

When demand remains unchanged, it is easily seen that any schedule computed after period H_1 is optimal. This scheduling technique loses its effectiveness when large fluctuations in demand are encountered from schedule to schedule. Under the assumption of reasonable accuracy in the forecast of demand, it will be shown that the Backward Solution Technique creates optimal schedules.

When computing a new schedule at time t, compute it from time 0. Then use the portion from t to H. The inventory at time t is the new initial inventory. The schedule from H_1 to H satisfies Assumption 2.2 and is optimal. Thus, the schedule from t to H is optimal. Let t_i

be the earliest period in this schedule in which product i has had completed at least one stage 2 job having a supplier job. The half-open interval $[t,t_i)$ is the <u>interval of accuracy</u> for the demand forecast for product i.

An increase in demand has two possible impacts on the schedule. There may be enough idle time in the necessary places to handle the added load; thus, Assumption 2.2 is maintained. Since the addition of a new job to a BST schedule can cause the jobs of lower-numbered products to be pushed to earlier periods, the alternative impact of an increase in demand is the creation of an infeasible problem. If an increase in demand is for a period earlier than H_1 and is feasible, it may cause a redefinition of H_1 to an earlier period. The same is true for period t_i.

A decrease in demand of one or less stage 2 jobs between schedules also has two possible impacts on the schedule. If the decrease in demand for product i applies to a period no earlier than t_i, it is easily shown that the affected jobs in the interval $[t,t_i)$ will maintain the same relative ordering as they advance in the schedule, as a result of the removal of the product i jobs. Thus, the schedule remains optimal. If the decrease in demand for product i occurs for a period earlier than t_i, the schedule may revert to a start-up mode.

In order for the scheduling method to maintain optimality after the start-up period, all decreases in demand for product i must occur no earlier than t_i and must be able to be satisfied by the cancellation of jobs which have a supplier job. This cancellation is accomplished when the demand forecast for product i is a lower bound on actual demand in the interval.

Conditions have been developed under which the BST gives an optimal solution. It was shown using worst case analysis that after an initial start-up period the method can accommodate normal changes in demand. In the next section the notation and assumptions will be extended to cover N stages in series. Lemma 2.3 and Theorems 2.1 and 2.2 will be extended to N stages in series.

2.7 The Extension to N Stages in Series

Up to this point the discussion has concerned a two-stage problem. A natural extension of this problem is to N stages in series. There is no intuitive difference between the two problems. All facets of the definition and solution of the new problem are the same, except that there are N stages to consider.

In the notation, the range of the stage indicator j is now from 1 to N. The incremental carrying cost coefficients, Ψ_i^2 and b_i^2, now become Ψ_i^j and b_i^j. The minimum number of supplier jobs in stage j for each job in stage j+1 is N(j,j+1) and is defined by

$$N(j,j+1) = \min_i \left\lfloor p_i^{j+1}/p_i^j \right\rfloor .$$

The functions $w_{i,t}^N(D)$ and $w_{i,t}^j(X^{j+1})$ replace $w_{i,t}^2(D)$ and $w_{i,t}^1(X^2)$, respectively. The Backward Solution Technique is unchanged, except that in Step 1 the stage counter j is initialized at N.

Assumption 2.1 (production rate) becomes

<u>Assumption 2.5</u>: The production rate $p_i^j \leq p_i^{j+1}$, for all i and for $j = 1, \ldots, N-1$.

Assumption 2.2 (supplier job) becomes

Assumption 2.6: Each stage j+1 job has at least N(j,j+1) stage j supplier jobs, for j = 1, ..., N-1.

Assumption 2.3 (machine availability) becomes

Assumption 2.7: The number of stage j machines

$$N_j \leq N_{j+1}N(j,j+1)\ , \quad \text{for } j = 1, \ldots, N-1.$$

Lemma 2.3 becomes Lemma 2.4 in which i1, i2, i3, i4, t1, t2, and t4 are defined exactly as they are for Lemma 2.3, except that they are in stages j and j+1 (Figure 2.10).

Lemma 2.4: Consider the Gantt chart of a feasible solution to an N-stages-in-series problem in which the first j stages each have a Dorsey solution and in which each stage j+1 job is scheduled so that there are no idle-machines between it and its relative deadline. If

(a) the jobs in the interval [t2+1, t4-1] of stage j+1 are of the product class i3,

(b) the (i1, j+1) job and the (i2, j+1) job are interchanged in the schedule, and

(c) Assumptions 2.5, 2.6, and 2.7 are met,

then there exists a feasible solution which is identical to the new solution in stage j+1 and has a Dorsey solution in the first j stages, for j = 1, ..., N-1.

Proof Outline: The proof is by induction on j. Note that after each interchange stages j+2 through N remain unchanged and, therefore, feasible.

TIME	t1	...	t2	...	t3-1	t3	...	t4-1	t4	...	t5
STAGE j	i2	...	i4	...	i4	i4	...	i1		...	
STAGE j+1		...	i2	...	i3	i3	...	i3	i1	...	

FIGURE 2.10. Portions of a solution for two adjacent stages of an N-stage problem.

For $j = 1$, Lemma 2.3 is the proof.

Make the induction assumption that Lemma 2.4 is true for $j = K>1$.

For $j = K+1$, make the interchange of the (i1, j+1) and (i2, j+1) jobs, as in the proof of Lemma 2.3. Then, as each pair of their stage j supplier jobs are interchanged to re-establish stage j feasibility, invoke the induction assumption to create feasible Dorsey solutions in stages 1 through j-1. After establishing a feasible solution in stage j, all that remains is to convert stage j to a feasible Dorsey solution while maintaining feasible Dorsey solutions in stages 1 through j-1. This is accomplished by means of a period by period conversion of stage j, like the one used on stage 2 in the proof of Theorem 2.1. After each interchange, invoke the induction assumption to maintain feasible Dorsey solutions in stages 1 through j-1. This concludes the induction and proves the lemma.

Q.E.D.

The results of Lemma 2.4 form the cornerstone of the proof of Theorem 2.3 (the successor of Theorem 2.1).

Theorem 2.3: If there exists a feasible solution to the N-stage problem and if Assumptions 2.5, 2.6, and 2.7 are met, then the Backward Solution Technique finds a feasible solution.

Proof Outline: The proof is by induction on N.

For $N = 2$, Theorem 2.1 is the proof.

Make the induction assumption that Theorem 2.3 is true for $N = K$.

For $N = K+1$, if there is a feasible solution, the first N-1 stages have, by the induction assumption, a feasible solution (found by the BST) which has a Dorsey solution in each stage. With this feasible

solution, all that remains is to convert stage N to a feasible Dorsey solution while maintaining feasible Dorsey solutions in stages 1 through N-1. This is accomplished by means of a period by period conversion of stage N to a feasible Dorsey solution, like the one used on stage 2 in the proof of Theorem 2.1. After each interchange invoke Lemma 2.4 to re-establish feasible Dorsey solutions in stages 1 through N-1. This provides a feasible solution to the N-stage problem which has a Dorsey solution in each stage. Since Dorsey solutions are unique, this is the solution found by the BST. Thus, the induction proof is complete.

Q.E.D.

To show that the BST finds an optimal solution, Assumption 2.4 becomes

Assumption 2.8: The inventory cost $b_i^j \leq b_{i+1}^j$, for $i = 1, \ldots, M-1$ and all j.

In order to show that the BST finds an optimal solution, it is necessary to prove an additional lemma. The lemma extends Lemma 2.4 to finding a new feasible solution which has an objective value at least as large as that of the original feasible solution. The lemma again refers to Figure 2.10.

Lemma 2.5: If the hypothesis of Lemma 2.4 and Assumption 2.8 are met, then there exists a feasible solution which has an objective value at least as great as that of the original feasible solution and which is identical to the new solution in stage j+1 and has a Dorsey solution in the first j stages, for $j = 1, \ldots, N-1$.

Proof Outline: The proof is by induction.

For $j = 1$, since the interchange and the determination of the Dorsey solution in the proof of Lemma 2.3 cannot decrease the objective value, the first induction step is proven.

Make the induction assumption that Lemma 2.5 is true for $j = K$.

For $j = K+1$, use the same proof as was used in the last step of the induction proof of Lemma 2.4. Invoke the induction assumption of the present proof in the place of the induction assumption of the proof of Lemma 2.4. The only other movement of jobs is by pairwise interchange. No interchange, by Assumption 2.8, decreased the objective value. Thus, the objective value of the feasible solution found during the induction is at least as great as that of the original feasible solution.

Q.E.D.

To show that the BST finds an optimal solution, Theorem 2.2 becomes

Theorem 2.4: If Assumptions 2.5, 2.6, 2.7, and 2.8 are met and if a feasible solution exists to the N-stage problem, then the Backward Solution Technique finds an optimal solution.

Proof Outline: The proof is by induction.

For $N = 2$, Theorem 2.2 is the proof.

Make the induction assumption that Theorem 2.4 is true for $N = K$.

For $N = K+1$, assume an optimal solution for the N-stage problem. By invoking the induction assumption, convert the first N-1 stages to a BST solution. This N-stage solution is still optimal and has Dorsey solutions in the first N-1 stages. Convert stage N to a Dorsey solution by using the same period by period conversion used in the proof of

Theorem 2.4. Instead of invoking Lemma 2.4 after each stage N interchange, use Lemma 2.5. The conversion results in a feasible solution which has a Dorsey solution in each stage and which has an objective value as great as that of the original optimal solution. Since this is a maximization problem and Dorsey solutions are unique. The BST would find this optimal solution.

Q.E.D.

It has been shown that the BST, under certain assumptions, finds an optimal solution, if a feasible solution exists. A more general production system is presented in the next section. Conditions under which the BST can find an optimal solution are discussed.

2.8 Further Extensions of the Backward Solution Technique

Until this section the discussion has concerned only problems in which the production system consisted of stages in series. To consider more complex systems, it is necessary to introduce the concept of a stage diagram. The stage diagram is a network representation of the production system. Each node in the diagram represents a production stage. The directed arcs connecting the nodes indicate a possible path a product can take through the system on its way to completion. In the stage diagram in Figure 2.11, the arcs show that a component produced in stage 3 must be used in at least one of the stages 6 through N-3. Finished products are output from both stages N-1 and N.

A supplier stage of stage j is any stage in the diagram from which stage j has an incoming arc. Let S(j) be the set of supplier stages of stage j. A consumer stage of stage j is any stage in the diagram to which stage j has an outgoing arc. Let C(j) be the set of all

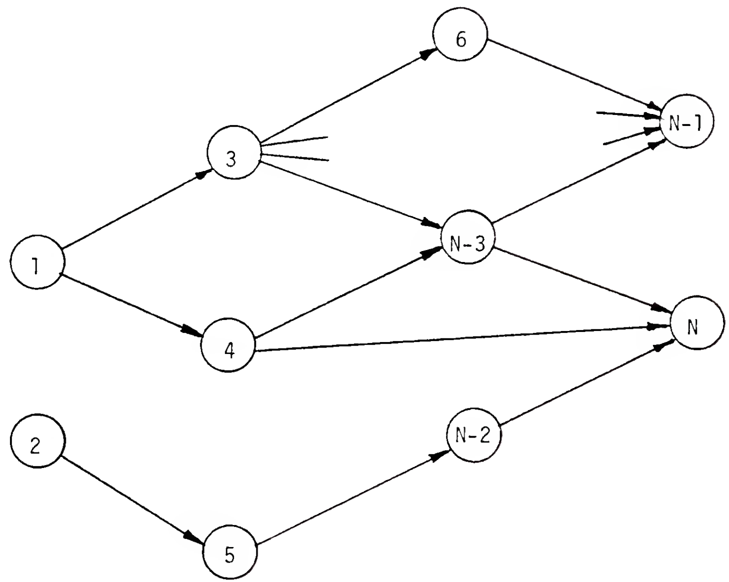

FIGURE 2.11. Stage diagram of a general production system.

consumer stages of stage j. In this terminology the production rate and supplier job assumptions are concerned with a stage and its supplier stages. Those two assumptions, along with the cost assumption, remain unchanged. The machine availability assumption, because a stage can have more than one consumer stage, becomes

Assumption 2.9: The number of machines

$$N_j \leq \min_{j1 \varepsilon C(j)} N_{j1} N(j,j1) .$$

A result of the supplier job assumption, which becomes more apparent for the general production system, is that any product produced in stage j must be produced in each of its supplier stages. However, a product produced in stage j is only required to be produced in at most one of its consumer stages. As an example from Figure 2.11, a product produced in stage N-3 must be produced in stages 3 and 4 but is only required to be produced in at most one of stages N-1 and N. Thus, there are only two possible paths for a product to follow through the production system represented in Figure 2.11. One path ends in stage N-1 and includes stages 1, 3, 4, and 6 through N-3. The other path ends in stage N and includes stages 1, 2, 3, 4, 5, N-3, and N-2.

The production system under consideration for the BST must be able to be represented by an acyclic, directed network. With the exception of the replacement of Assumption 2.7 by Assumption 2.9, the feasibility and optimality theorems and their proofs are essentially unchanged. The BST would solve the stages of the system in Figure 2.11 in the order of their decreasing stage numbers.

A general production system was described. The assumption alterations necessary to accommodate the system were explained. Finally, it was noted without proof that with these changes the BST finds an optimal solution to the general production system, if a feasible solution exists.

2.9 Conclusion

An extension of a single-stage, multi-machine, multi-product scheduling problem into a multi-stage problem was discussed in this chapter. The two-stage problem was considered first. A method, called the Backward Solution Technique, was introduced which solves each stage with a greedy algorithm developed by Dorsey [12]. It was found that when the bottleneck occurs in the initial production stage (the production rate and machine availability assumptions) and when production start-up effects are over (the supplier job assumption), the BST finds a feasible solution, if one exists. With the addition of the cost assumption, the BST finds an optimal solution. It was shown by counter-example that relaxation of the assumptions can lead to failure to find the desired solution. Practical application of the BST was discussed. It was found that after a start-up period and with reasonably accurate demand forecasts, the production system met the requirements for the effective use of the BST.

After the two-stage problem an extension to N stages in series was presented. In the final extension, the production system formed a general, directed, acyclic network. With few basic changes, the assumptions, lemmas, and theorems remained the same for each of the extensions.

The next logical avenue of exploration is the case in which $p_i^j \geq p_i^{j1}$ for $j1 \varepsilon C(j)$ and all j. This moves the production bottleneck to the last stage. This problem is considered in the next chapter.

CHAPTER 3

A POSTERIOR BOTTLENECK PROBLEM

3.1 Introduction

In Chapter 2 the frontal bottleneck problem was discussed. The cause of the frontal bottleneck was that the production rate for each product was a nondecreasing function of its level within the network of stages. As a result of the production rate at least one supplier job exists in each supplier stage for every job in a given stage. Such a property along with properties on the inventory carrying cost and the number of machines in successive stages allowed the development of a single-pass algorithm for an optimal solution.

In this chapter the case in which the production rate for each product is a nonincreasing function of its level within the network of stages is dealt with. This causes the existence of no more than one supplier job in each supplier stage for every job in a given stage. In fact, some jobs may have no supplier jobs at all and may depend upon receiving their input from the supplier job of some other job in their stage. This creates a potential production bottleneck in the final stages of production.

In this chapter it is shown that under certain conditions this problem can be reformulated as a single-stage, multi-machine, multi-product scheduling problem in which there are precedence constraints among some of the jobs. Each job will have linear deferral costs and an availability time which may differ from job to job.

It may appear that this is a relatively easy problem to solve. However, Lenstra [30] has shown that the problem is NP-complete. This caused Elmaghraby and Sarin [13] to examine the bounds on a heuristic for a relaxed version of the problem.

Other papers deal with further relaxations of the problem in which each job has the same availability time and there is a single machine. Horn [23] solves the problem in which the precedences are sets of arborescences (forests). Adolphson and Hu [1] solve the problem for an arborescence in worst case time O(n log n) and shorten Horn's proof. Lawler [29] solves the problem in which the precedences involve parallel series of jobs and realizes a worst case time of O(n log n). Sidney [37] develops an algorithm which decomposes and solves general, acyclic networks of jobs.

Hodgson and Loveland [21,22] address a multi-machine problem which has similar structure but minimizes the completion time of the latest job. Bartholdi, Martin-Vega, and Ratliff [2] survey parallel processor scheduling problems.

3.2 Formulation

In this section the posterior bottleneck case of the multi-stage, multi-machine, multi-product scheduling problem is formulated. The conditions under which the problem can be reformulated as a single-stage problem are developed. Since even the single-stage problem is difficult, there is a discussion of solutions to relaxed versions of the problem in preparation for the heuristics of Chapter 4.

Consider a multi-stage, multi-machine, multi-product scheduling problem in which all the stages in the whole production system must form an assembly network. An example of such a system is in Figure 3.1A.

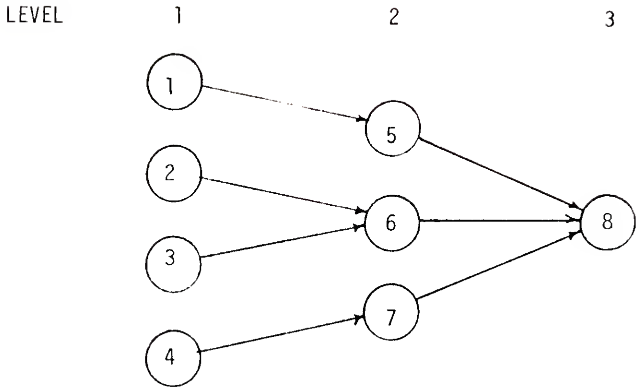

Figure 3.1A. General product system.

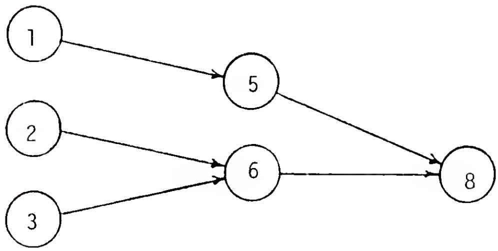

Figure 3.1B. Stage diagram for first product.

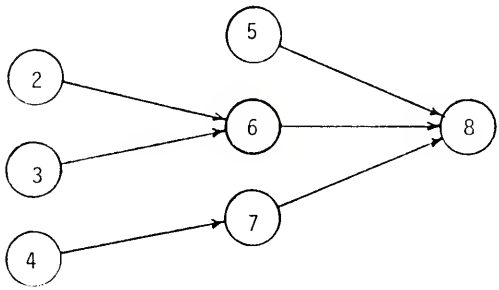

Figure 3.1C. Stage diagram for second product.

Each node is a production stage and is in one of L production levels. A product does not have to go through each stage. In this example there are two products. The first product only uses the stages shown in Figure 3.1B; the second product uses the nodes in Figure 3.1C.

The notation and terminology are the same as those of Chapter 2. The mathematical programming formulation is

$$\text{Max} \sum_{i=1}^{M} \sum_{j=1}^{N} \sum_{t=1}^{H} tb_i^j \, x_{i,t}^j \tag{3.1}$$

s.t.

P3.1

$$\sum_{i=1}^{M} x_{i,t}^j \leq N_j \,, \qquad j = 1, \ldots, N \text{ and } t = 1, \ldots, H \tag{3.2}$$

$$p_i^j \sum_{k=1}^{t} x_{i,k}^j - p_i^{j1} \sum_{k=1}^{t+1} x_{i,k}^{j1} \geq -I_i^j(0) + \delta_{tH} \, I_i^j \,, \tag{3.3}$$

$i = 1, \ldots, M$, $t = 1, \ldots, H$,

$j1$ a consumer stage of stage j,

and j not in level L.

$$p_i^{j1} \, x_{i,1}^{j1} \leq I_i^j(0) \,, \qquad i = 1, \ldots, M, \ j1 \text{ a consumer stage} \tag{3.4}$$

of stage j, and j not in level 1.

$$p_i^j \sum_{k=1}^{t} x_{i,k}^j \geq \sum_{k=1}^{t} d_{i,k} - I_i^j(0) + \delta_{tH} \, I_i^j \,, \tag{3.5}$$

$i = 1, \ldots, M$, $t = 1, \ldots, H$,

and j in level L.

$x_{i,t}^j \geq 0$, integer.

(Note: $x_{i,H+1}^j \equiv 0$ and $p_i^j \equiv 0$ for $i \notin G(j)$.)

As in Chapter 2, the objective function (3.1) is the nonconstant portion of the inventory carrying cost. Since the cost is nondecreasing over time, the jobs are forced to be scheduled as late as possible. Constraint (3.2) ensures that no more jobs are scheduled in a period than there are machines for that stage. The purpose of constraint (3.3) is to force the production of the first t periods of stage j to be great enough to supply that part of the input requirements of the first t+1 periods of stage j1 production which is not supplied by stage j initial inventory. The same constraint causes the desired level of final inventory for stage j to be reached. Constraint (3.4) shows that production in the first period of stage j1 is limited by the initial inventory of stage j. The effect of constraint (3.5) is the same as that of constraint (3.3), except that the demand for finished goods is satisfied instead of the input requirements of a consumer stage.

Consider problem P3.1 when

$$p_i^j \geq p_i^{j1} \text{ for } j1 \text{ a consumer stage of } j.$$

Under these circumstances an (i,j) job produces at least as many units as an (i,j1) job requires. Assumption 2.2 of Chapter 2 no longer is satisfied. Each (i,j1) job has no more than one supplier job in stage j. The production bottleneck is in the final stages. As a result, the analysis of Chapter 2 does not apply here.

For two stages in series, if the supplier stage has as many machines as the consumer stage, the supplier jobs are able to be scheduled at their relative deadlines. Figure 3.2 shows two such stages, each of which has two machines. The product 2 job in period 2 of stage 1 supplies the input for both product 2 jobs in stage 2.

Time	1	2	3	4
Stage 1	1	1	1	
	1	2		
Stage 2		1	1	1
		1	2	2

Figure 3.2 Two stages in series in which $p_1^1 = p_1^2$ and $p_2^1 = 2p_2^2$.

Its consumer job is the product 2 job in period 3 of stage 2. Thus, the stage 1 job has its relative deadline at period 2. Since each stage 2 job has at most one supplier job in stage 1 and since stage 1 has at least as many machines as stage 2, every feasible schedule can have the jobs of stage 1 at their relative deadlines. In particular, the optimal solution has the jobs of stage 1 at their relative deadlines. The position of the consumer job completely determines that of its supplier.

It is easily seen that for a set of stages in series, in which the number of machines in a stage is at least as great as the number in its consumer stage, the positions of the jobs in the final stage of an optimal solution completely determine the positions of the jobs in all the earlier stages. This leads to the following theorem.

Theorem 3.1: If $p_i^j \geq p_i^{j1}$ and $N_j \geq N_{j1}$ for $j \in S(j1)$ and the stages of the production system form an assembly network, the positions of the jobs in the final stage of the optimal solution of problem P3.1 completely determine the optimal positions of the jobs in the earlier stages.

Proof: By the above argument, the last stage of the optimal schedule determines the optimal position of all the jobs in any path (i.e., series of stages) through the network. Since each stage in a given level of the assembly network is independent of every other stage in that level, the terminal stage of the optimal schedule determines the optimal position (one period earlier than their consumer jobs) of all the jobs in every path through the network.

Q.E.D.

By Theorem 3.1 optimally solving this multi-stage problem is equivalent to finding the schedule of the stage in level L of its optimal solution. In every optimal solution each job attempts to be at its deadline. Since there are always enough machines in the supplier stages for that purpose, each optimal solution to the problem has the jobs in stages of level less than L at their relative deadlines. Because of this, all the supplier jobs and consumer jobs of every job can be designated before any solution procedure starts.

All of this requires, without loss of generality, an ordering among the jobs of each product in the last stage (level L). The ordering does not change the optimal schedule. It merely numbers the jobs ahead of time and ensures that they are in order in all, including the optimal, schedules. The effect of the ordering, for some stage j in level L, is to ensure that the first (i,j) job can be scheduled no later than the second (i,j) job, etc. An example of this appears in Figure 3.3A.

Figure 3.3A shows a production system consisting of three stages in series, each having one machine. There is a single product which uses all three stages. Figure 3.3A is a Gantt chart in which the numbers are job numbers for that stage of the single product. The production rates, as shown in Figure 3.3A, are such that the stage 2 jobs in periods 2 and 3 are supplier jobs of jobs 1 and 2, respectively, in stage 3. The job in stage 1 is a supplier of job 1 in stage 2. Actually, job 1 in stage 1 supplies the input to both jobs 1 and 2 in stage 2. Thus, it must be scheduled earlier than the earlier of those two jobs in stage 2. A precedence constraint would require job 1 to be performed no later than job 2 in stage 3. If, in addition to the

Time	1	2	3	4
Stage 1	1			
Stage 2		1	2	
Stage 3			1	2

$p_1^1 = 2,\ b_1^1 = 5$

$p_1^2 = 1,\ b_1^2 = 20$

$p_1^3 = 1,\ b_1^3 = 30$

Figure 3.3A. Solution for single product, 3 stages.

Time	1	2	3	4
Stage 3			1	2

$p_1^3 = 1,\ b_{1,1}^3 = 55,\ b_{1,2}^3 = 50$

Figure 3.3B. Solution for Figure 3.2A reduced to 1 stage.

precedence, the relative deadlines of the jobs are observed, the jobs in each stage will be ordered by their job numbers.

If the above ordering is kept, a job in some level k(<L) is in period t if and only if its consumer job is in period t+1. Thus, moving a job in a stage in level L causes all of its supplier jobs to move with it by the same number of periods. It is reasonable, therefore, to add the cost of the level k job to that of its consumer job and remove the level k job from the problem. If this is done for all level k jobs and for all levels k(<L), starting at the first level, this multi-stage problem becomes a single-stage problem which has precedence relationships among the jobs of the same product. Figure 3.3B demonstrates the reduction of the problem of Figure 3.3A from three stages to a single stage problem having cumulative costs and a precedence constraint between jobs 1 and 2.

The following theorem formally states the conditions for this stage reduction.

Theorem 3.2: If (a) $p_i^j \geq p_i^{j1}$ and $N_j \geq N_{j1}$ for $j \in S(j1)$, (b) the production system is an assembly network, (c) the production stages of each product are connected by arcs, (d) and a series ordering is placed on the jobs of the same product within the terminal stage, then problem P3.1 can be reformulated as a single-stage problem which has precedence constraints among the jobs of the same product.

Proof: The proof was established in the preceding discussion.

Q.E.D.

In this new single-stage problem the jobs are distinguishable because of the precedence constraints. As a result, the demand and the time at which it occurs can be translated into due dates for the individual jobs.

In order to conform with similar problems in the literature, the time frame of the single-stage problem will be reversed. This translates the problem into one of minimization, thus forcing the jobs as early as possible. The due dates become availability times.

The notation for the new problem has $c_{i,k}$ as the combined cost coefficient of suppliers for the k^{th} job of product i. The constant M_i is the number of jobs of product i. Also, M, H, and N are the number of products, the horizon, and the number of machines, respectively. The term $a_{i,k}$ is the availability time of the k^{th} job of product i. The decision variable $x_{i,k,t}$ is 1 if the k^{th} job of product i is scheduled in period t and is 0 otherwise. The new problem has the following mathematical programming formulation for the terminal stage:

$$\min \sum_{i=1}^{M} \sum_{k=1}^{M_i} \sum_{t=a_{i,k}}^{H} tc_{i,k}x_{i,k,t} \tag{3.6}$$

s.t.

P3.2

$$\sum_{t=a_{i,k}}^{H} x_{i,k,t} = 1,\ i = 1, \ldots, M \text{ and } k = 1, \ldots, M_i \tag{3.7}$$

$$\sum_{i=1}^{M} \sum_{k=1}^{M_i} x_{i,k,t} \leq N,\ t = 1, \ldots, H \tag{3.8}$$

$$x_{i,k,t} - \sum_{r=t}^{H} x_{i,k+1,r} \leq 0,\ i = 1, \ldots, M,\ k=1, \ldots, M_i-1, \quad \text{and } t = a_{i,k}, \ldots, H \tag{3.9}$$

$$x_{i,k,t} = 0 \text{ or } 1.$$

(Note: $x_{i,k,t} = 0$ for $t < a_{i,k}$.)

The objective function of P3.2, represented by (3.6), is a linearly increasing function over time and, thus, forces the jobs to be scheduled as early as possible. Constraint (3.7) ensures that each job is scheduled exactly once and in its interval of availability. Constraint (3.8) limits the number of jobs in a period to the number of available machines. Constraint (3.9) is a precedence constraint which maintains the ordering relationship among the jobs of the same product.

As is pointed out in the literature review, Lenstra shows this problem is NP-complete. Therefore, an attempt to find a way to solve it efficiently would appear fruitless. A more promising approach seems to be to find a good heuristic. A place to look is at optimal solution methods for relaxed versions of P3.2.

Clearly, P3.2 without the constraints (3.9) is the well known transportation problem. It is easily and efficiently solved. However, because the cost coefficients may fluctuate so greatly from job to job, it is doubtful that a network solution would show much resemblance to an optimal solution to P3.2.

If the number of machines, N, is dropped to one and the availability times become zero, the problem is well solved. Sidney has the most generally applicable method for optimally solving the single-machine problem. His method is the basis for one of the heuristics of the next chapter.

3.3 Conclusion

Another case, the posterior bottleneck, of the multi-stage, multi-machine, multi-product scheduling problem was examined in this chapter.

The bottleneck exists when the production rate for any given product is a nonincreasing function of the level of the stage to which it applies ($p_i^j \geq p_i^{j1}$ for $j \in S(j1)$). Unlike Chapter 2, no restrictions were placed on the cost coefficients of the jobs. The production system was an assembly network of stages.

It was explained that numbering the jobs of each product of the terminal stage and requiring the jobs to be scheduled in order does not cause a loss in the generality of the solution. It merely allows the identification of a job's supplier and consumer jobs before the solution procedure begins. As a result, it was shown that, with the proper lower bound on the number of machines in the stages outside level L, the problem can be reformulated as a single-stage problem having serial precedence constraints among the jobs of the same product.

Solution approaches for the problem and its relaxed versions were discussed at the end of the second section and in the literature review of the first section of the chapter. It was decided that since the problem is NP complete, heuristics, one of which will be based upon Sidney's algorithm for the single-machine relaxation of the problem, will be developed to solve the problem. The two heuristics of the next chapter include an efficient method for the multiple interchange of jobs in a feasible schedule. An enumeration algorithm is developed which takes advantage of many properties of the problem to improve its efficiency. The enumeration algorithm is used to test the accuracy of the heuristics.

The results of 200 randomly generated problems are presented in the latter part of the chapter. Statistics are developed to measure the efficiency and accuracy of the heuristics.

CHAPTER 4

HEURISTICS AND TESTING

4.1 Introduction and Formulation

Consider the problem of scheduling M products on N identical machines over a horizon of length H. Each product i has M_i unit-duration jobs. Each job k of product i has a deferral cost of $c_{i,k}$ units per period. In addition, each job k has an availability time $a_{i,k}$. Within the products the jobs are ordered by serial precedence constraints (job number). For product i, job k can be scheduled no later than job k+1. Let $x_{i,k,t}$ be the 0-1 decision variable which is 1 if job k of product i is scheduled in period t and 0 otherwise.

The objective of the problem is to schedule all the jobs within the horizon while maintaining the precedence constraints and minimizing the deferral cost. The mathematical programming formulation of the problem is

$$\min \sum_{i=1}^{M} \sum_{k=1}^{M_i} \sum_{t=a_{i,k}}^{H} t c_{i,k} x_{i,k,t} \tag{4.1}$$

s.t.

P4.1

$$\sum_{t=a_{i,k}}^{M} x_{i,k,t} = 1, \quad i = 1, \ldots, M \text{ and } k = 1, \ldots, M_i \tag{4.2}$$

$$\sum_{i=1}^{M} \sum_{k=1}^{M_i} x_{i,k,t} \leq N, \quad t = 1, \ldots, H \tag{4.3}$$

$$x_{i,k,t} - \sum_{r=t}^{H} x_{i,k+1,r} \leq 0, \; i = 1, \ldots, M; \; k = 1, \ldots, M_i - 1; \text{ and } t = a_{i,k}, \ldots, H \tag{4.4}$$

$x_{i,k,t} = 0$ or 1.

(Note: $x_{i,k,t} \equiv 0$ for $t < a_{i,k}$.)

The problem P4.1 without constraint (4.4) can be solved as a network flow problem. Thus, problem P4.1 can be viewed as a network problem having additional complicating constraints. Problem P4.1 has been shown to be NP-complete. For that reason another approach must be taken to the solution of the problem.

The following sections are used to develop and test two heuristics augmented by a multiple interchange method. An enumeration algorithm is developed to test the accuracy of the heuristics. This and the heuristics are combined as subroutines into a Fortran program for the purpose of testing on randomly generated problems. The final section describes the testing and results.

4.2 The Heuristics

Since the problem P4.1 is a very difficult problem to solve, two heuristics, SCHED1 and SCHED2, have been developed and programmed in Fortran IV. To augment the heuristics, a method of performing multiple interchanges on the jobs in a schedule has also been programmed. The method is applied to the solution found by each heuristic in an attempt to improve it. This section describes the heuristics and the multiple interchange method applied to them.

Define a feasible substring of a job string to be a substring which contains the first job on that job string and consecutive jobs which are available to successive machines. Use Figure 4.1 as an example and let there be two machines in each period. Also, let the next available machine be the first machine in period 1. There are three feasible

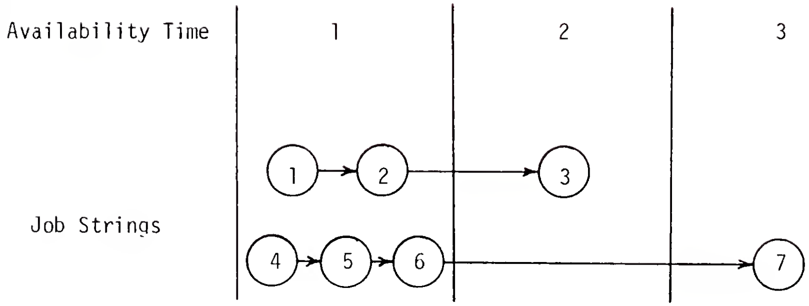

Figure 4.1. Availability times and precedences for a two product problem.

substrings of the first job string. Job 1 is a feasible substring because it is available to that first machine in the first period. Jobs 1 and 2 form a feasible substring because job 2 is available for the second machine of period 1 while machine 1 does job 1. Jobs 1, 2, and 3 form a feasible substring. Since jobs 1 and 2 can occupy the two machines in period 1, the first machine in period 2 becomes the next available machine. Job 3 is available for it. For similar reasons, other feasible substrings are job 4; jobs 4 and 5; and jobs 4, 5, and 6. Jobs 4 through 7 do not comprise a feasible substring. After jobs 4 through 6 are assigned to the machines of period 1 and the first machine of period 2, the second machine of period 2 becomes the next available machine. Job 7 is not available for that machine and cannot be included in the substring. If, however, the jobs in Figure 4.1 had machine 2 of period 1 as the next available machine, machine 1 of period 3 would be the next available machine after scheduling jobs 4 through 6. Job 7 is available for that machine. Thus, jobs 4 through 7 would comprise a feasible substring.

The first heuristic (called SCHED1) is based on a measure of the potential penalty of not scheduling a given job on the first available machine. The calculation is made for the first job, if available, of every job string. Considering the job for scheduling on the second available machine, not the first available machine, the longest feasible substring of that job string is determined. Analysis is performed as if the substring were to be scheduled next, starting on the second available machine. The jobs of interest are those which fall on the first machine in a period. Either their availability time is that period or they would be scheduled in the preceding period, if the string's first job were scheduled on the first available machine. If that job on the first

machine of the period is at its availability time, neither it nor its successor jobs are considered in this analysis. If, however, it is not at its availability time, the job has been forced one period later by scheduling the string's first job on the second available machine. This job and all like it would cause the penalty incurred by not scheduling the string's first job on the first available machine. Thus, the penalty cost on the first job of the string is defined as the sum of the deferral costs of all such jobs which would be forced a period later if the string's first job is scheduled on the second available machine.

The analysis is done for every job string. The available first job having the greatest penalty is actually scheduled on the first available machine and removed from its job string. The analysis is then repeated with the new first available machine for all the job strings until all jobs have been scheduled.

The second heuristic (called SCHED2) is an adaptation of Sidney's algorithm [37] for the one-machine problem in which all jobs have the same availability time. Each product is represented by a string of jobs in series. Their order on the string defines the precedences between them.

As an example, consider the strings in Figure 4.1. Each node is a job. Jobs 1, 2, 4, 5, and 6 become available in period 1. Job 1 has precedence over job 2 which means job 2 can be scheduled no earlier than job 1. Each job string corresponds to a product. Thus, jobs 1, 2, and 3 produce one product and jobs 4 through 7 produce another.

SCHED2 generates all the feasible substrings for a given next available machine. For each such substring the ratio of the cumulative

deferral costs to the number of member jobs is determined. The feasible substring having the largest ratio is the candidate for scheduling. If there is a tie between substrings of the same string, the first substring calculated is used. In order to conform with Sidney's tie breaking rule, all ties should go to the shortest substring.

After the final candidate substring has been selected, its jobs are scheduled on the earliest available machines. This creates a new next available machine. The jobs which have just been scheduled are deleted from their job string. New feasible substring and ratio calculations are then made to determine the next substring to be scheduled. If no feasible substrings exist but some jobs remain to be scheduled, the first machine in the next period becomes the next available machine. This method is repeated until all the jobs have been scheduled.

4.3 A Multiple Interchange Method

After a schedule has been completed by SCHED1 or SCHED2, it may be beneficial to try to improve the solution by interchanging two or more jobs. In the present case, two-job, three-job, and four-job interchanges are used. What follows is a discussion of the mechanics of applying this concept to the problem at hand.

A two-way interchange of jobs k1 and k2 in a schedule would replace k1 on its machine with job k2 and place job k1 on the machine formerly occupied by k2. An example of a two-way interchange is presented in Figure 4.2B in which jobs 1 and 2 are interchanged. A three-way interchange involves three jobs switching places in a schedule, as in Figure 4.2A.

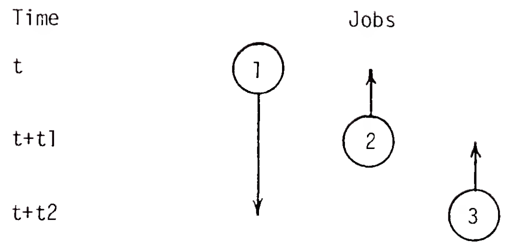

Figure 4.2A. Standard 3-way interchange

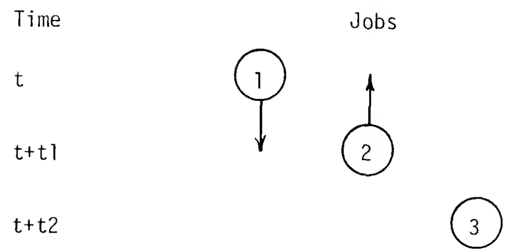

Figure 4.2B. Standard 3-way interchange--first step

Multiple interchanges of jobs are used to improve the heuristic solutions. For experimentation, the interchange method accommodates two, three, and four-way interchanges.

The interchange method has the following form for a k-way interchange:

<u>Step 1</u>. Set i = 2 and make all i-way interchanges of jobs which improve the solution.

<u>Step 2</u>. If $i < k$, set i = i+1. Otherwise, stop.

<u>Step 3</u>. If an i-way interchange which would improve the solution exists, make it and go to Step 1. Otherwise, go to Step 2.

By the time the interchange method has been completed, there are no two-way through k-way interchange improvements to be made in the final solution.

The form of a two-way interchange needs no further explanation. The standard three-way interchange is demonstrated in Figure 4.2A. Job 1 replaces job 3 in period t+t2. Job 3 replaces job 2 in period t+t1 which, in turn, replaces job 1 in period t. The only other three-way interchange (a mirror image) occurs when job 1 replaces job 2 in period t+t1; job 2 replaces job 3 in period t+t2; and job 3 replaces job 1 in period t.

The standard three-way interchange generates four cases which improve the solution. Only a subset of those cases needs to be considered, as is shown later. Of those four cases, three can be decomposed into two-way interchanges which improve the solution at least as much. Thus, only one case requires an actual three-way interchange be performed to accomplish its goals. (As will be seen in later discussion, this can result in computational savings.)

Case 1 involves either a precedence constraint between jobs 2 and 3 or a higher cost $c_{i,k}$ for job 2 than job 3. In this case both jobs 2 and 3 have higher costs $c_{i,k}$ than job 1. Therefore, the interchange can be decomposed into the two-way interchange of Figure 4.2B, followed by that of Figure 4.2C. Both two-way interchanges improve the solution.

Case 2 involves no precedence constraint between jobs 2 and 3 and a higher cost for job 3 than job 2. Since job 3 also has a higher cost than job 1, the two-way interchange of Figure 4.2D would be more advantageous and would save the inevitable interchange of jobs 2 and 3.

Case 3 involves a cost for job 3 no greater than that of job 1. In this case, the two-way interchange of Figure 4.2B accomplishes the improvement and saves a possible interchange of jobs 1 and 3.

Case 4 is the genuine three-way interchange of Figure 4.2A. It requires a precedence constraint between jobs 2 and 3 and a cost no higher for job 2 than job 1. It is this case and its mirror image which the three-way interchange routine is programmed to detect.

Where there are only two standard three-way interchanges (including mirror images), the four-way interchange has five standard forms (including mirror images). Two of those and part of a third are, unconditionally, sets of pairwise interchanges. Other parts of that third form are a combination of a three-way and two-way interchange. The result is that instead of programming five different searches, only two full searches and a subset of a third are necessary. It appears that these forms have the smallest number of potential interchanges.

The result of all this is that the computational expense which is expected in combinatoric problems having this complexity can be

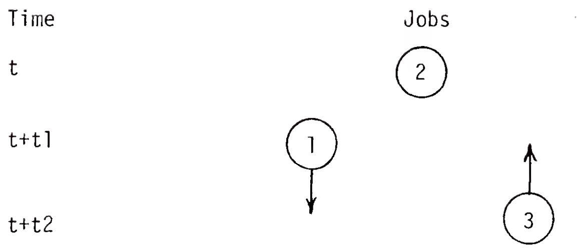

Figure 4.2C. Standard 3-way interchange--second step

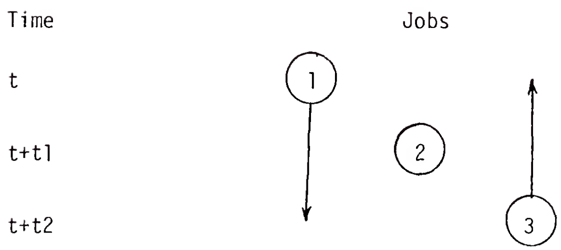

Figure 4.2D. Standard 2-way interchange

alleviated for these multiple interchange problems. With some insight into the problem being solved and some analysis of the component parts of the interchanges, the more complex interchanges can be decomposed into lower order, profitable interchanges which are more easily detected and performed.

Two heuristics and a multiple interchange method are described in these sections. They are used to solve problem P4.1, which has the form of a network problem with additional complicating constraints.

The following sections will cover an enumeration algorithm which takes advantage of some useful properties of the problem and testing of the heuristics. The enumeration algorithm is used in the testing in order to compare the heuristics against optimality.

4.4 The Enumeration Algorithm

An enumeration routine is needed to test the accuracy and efficiency of the heuristics. In order to make testing of a large number of problems feasible, the enumeration algorithm must be as efficient as possible.

The enumeration algorithm is developed in this section after the presentation of some properties of the problem which are useful in paring down the search tree. Separability and a relaxed version of the problem which acts as a lower bound are presented first.

The precedence constraints used in the problem are of a looser form than are normally seen. Rather than constraining job 2 to being done after job 1, they cause job 2 to be done no earlier than job 1. These looser precedences lead to a very useful property.

Property 4.1: A machine is empty in the heuristic solution iff it is empty in the optimal solution.

Proof: The proof is by contradiction. It will be sufficient to look at the earliest instance in which empty machines in the heuristic and optimal solution do not coincide. Clearly, if no such earliest instance occurs, the proof is complete. As a matter of terminology, for job 1 and 2 scheduled in the same period, job 1 is earlier than job 2 if it is on a lower numbered machine.

Assume that the optimal solution has the earliest unmatched empty machine. There must exist a job in the heuristic schedule which is no later than the empty machine but is later than that machine in the optimal schedule. Thus, its availability time and those of its predecessors are no later than the period of the optimal solution's empty machine. Consider the job's earliest predecessor which is scheduled later than the empty machine in the optimal solution. This job could be feasibly shifted into the empty machine resulting in an improvement of the objective function. This contradicts the optimality of the solution, and shows that the optimal solution will not have the earliest unmatched empty machine.

Assume that the heuristic solution has the earliest unmatched empty machine. By the very nature of the heuristic (either SCHED1 or SCHED2), that machine would be full unless each job scheduled later than it is not available until after that period. However, there must exist a job in the optimal solution which is no later than the empty machine but is later than that machine in the heuristic schedule. Thus, this job must be available by the period of the empty machine. This contradicts this solution being a product of the heuristic. Thus, the heuristic solution will not have the earliest unmatched empty machine.

There can be no earliest unmatched empty machine.

Q.E.D.

Clearly, empty machines in the optimal solution delineate problem sections which could be solved independently. The problem is separable. Property 4.1 shows that these empty machines and separable sections can be identified by the heuristics.

A separable section of the single-stage problem considered here has a property which results from the precedence constraints. No job can be scheduled on the first machine in its predecessor's availability time period. This property would be redundant as a constraint. However, consider it as a constraint, which creates a revised problem.

Clearly, the revised problem has the same optimal solution set as the original problem. Thus, relaxing the precedence constraints but maintaining the new constraint supplies a lower bound on the original problem. To break ties, jobs with equal deferral costs are scheduled by increasing job number. A desired characteristic of a lower bound is that it also supplies a lower bound on the completion of any partial optimal schedule.

In solving this relaxed, revised problem, the machines within a time period are scheduled in the order of their increasing number. The effect is that in period t the job on machine i has a deferral cost no less than that of the job on machine i+1, unless it violates the new constraint of the revised problem.

Property 4.2: For any partially completed optimal schedule, the optimal solution to the relaxed, revised problem provides a lower bound on the value of the final portion of the optimal schedule.

Proof: In this proof an arbitrary machine and period will be chosen in the optimal schedule. This machine will be considered the last scheduled machine in the partially completed optimal schedule. In the next step another schedule, based on the precedence constraint relaxation, is developed. This second schedule consists of an optimal solution to the relaxation of the partially completed optimal schedule and an optimal solution to the relaxation of the remainder of the optimal schedule. By a series of pairwise interchanges, the second schedule will be converted to a schedule containing all jobs in the same sections as those in the optimal solution to the relaxed, revised problem. It will be shown that the portion of the optimal relaxed, revised solution which succeeds the arbitrary machine chosen above provides a lower bound to the value of the corresponding portion of the optimal schedule.

Consider an optimal schedule S to the problem. Next, arbitrarily choose machine k in period t. Consider the jobs in the first t-1 periods and the first k machines of period t of S and call that schedule S_1. The remainder of schedule S is called schedule S_2. Solve S_1 and S_2 as relaxed, revised problems having optimal schedules S_{r1} and S_{r2}, respectively. Clearly, S_{r1} and S_{r2} are lower bounds for S_1 and S_2, respectively.

Let S_r^* be an optimal schedule for the relaxed, revised problem corresponding to the whole schedule S. S_{r1}^* and S_{r2}^* are the two sections of S_r^* which cover the same periods and machines as S_{r1} and S_{r2}, respectively. If S_{r1} and S_{r1}^* contain the same jobs, $S_{r1} = S_{r1}^*$ and $S_{r2} = S_{r2}^*$.

A property of S_r^* can be easily seen. Let q1 be any job in S_r^* and let job q2 be any job in S_r^* which is between q1 and the first machine available to q1. The property is that the deferral cost of q1 is no greater than that of q2. The same property holds if q1 and q2 are both

in S_{r1} or S_{r2}. The same property, clearly, holds for S^*_{r1} and S^*_{r2}. The next step is to convert S_{r1} and S_{r2} into schedules having the same jobs as S^*_{r1} and S^*_{r2}, respectively. This is done by converting S_{r1} into S^*_{r1}.

Find the earliest machine in S_{r1} which contains a job different from the one in that machine in S^*_{r1}. Let that be machine k1 in period t1. Let job j1 be in machine k1 in S_{r1} and job j2 in S^*_{r1}. Thus, j2 is later than j1 in S_{r1} or is in S_{r2}. Clearly, the first machine available to job j1 is no later than machine k1 in period t1. The same is true for job j2. Since j2 is earlier than j1 in S^*_r, j2 has a deferral cost no less than that of j1.

If job j2 is in S_{r1}, then all the jobs, including j1, between it and job j1 have at least as high a deferral cost as job j2. Thus, j1 and j2 have the same deferral cost. They can be interchanged in S_{r1} forming S^1_{r1} with no change in objective value. S_{r2} becomes S^1_{r2}.

If instead, job j2 is in S_{r2}, let it be in machine k2 in period t2. Move job j2 forward into machine k1 in period t1. Since the cost of j2 is no less than that of j1, no job in S_{r1} which is later than j1 and is available at machine k1 in t1 is more costly than j2. Move job j1 later in S_{r1} until it reaches a job less costly than it. Replace that job with j1 and, in turn, move it later in S_{r1} until a cheaper job is found. Continue this string of exchanges until the moving job moves later than machine k in period t (into S_{r2}). Call this job jn. Since job jn is less expensive than j2, move job jn into machine k2 in period t2. The net result in S_{r2} is that job jn replaces j2, causing a decrease in the value of S_{r2}. Make a pairwise interchange between job jn and the next later job available at the time and machine of jn until a cheaper job is reached. No such interchange increases the value of S_{r2}. The new schedules created from

S_{r1} and S_{r2} are S^1_{r1} and S^1_{r2}, respectively. S^1_{r2} is a lower bound on S_{r2} and, thus, on S_2. S^1_{r1} and S^1_{r2} also have the property that no job between job j and its first available machine is less costly than job j.

Repeated use of the preceding switching method, starting with S^1_{r1} and S^1_{r2} in place of S_{r1} and S_{r2}, will result in a pair of schedules S^n_{r1} and S^n_{r2} in which S^n_{r1} and S^n_{r2} have the same jobs as S^*_{r1} and S^*_{r2}, respectively. S^n_{r2} will be a lower bound for S_{r2} and, therefore, for S_2. By its optimality, S^*_{r2} is a lower bound for S^n_{r2} and, therefore, for S_2. This proves the property.

Q.E.D.

As a result of Property 4.2, a procedure which solves the relaxed, revised version of the schedule (called LOWBND) has the desired characteristics of a lower bound for an enumeration algorithm.

Useful properties of the problem have been presented and incorporated into a lower bounding technique. The special form of the precedence constraints, the problem separability, and LOWBND will be used extensively in the enumeration algorithm. Other important properties will be developed for the algorithm.

Consider the tree in Figure 4.3. It corresponds to the search tree of the enumeration algorithm for three products. At each stage in the algorithm (or level in the tree) there is a choice to be made between three alternatives. The choice is which product to schedule next. After that choice the first available job of that product is scheduled on the machine which corresponds to this level of the tree. The machines correspond to levels in order of increasing period number and increasing machine number within the period.

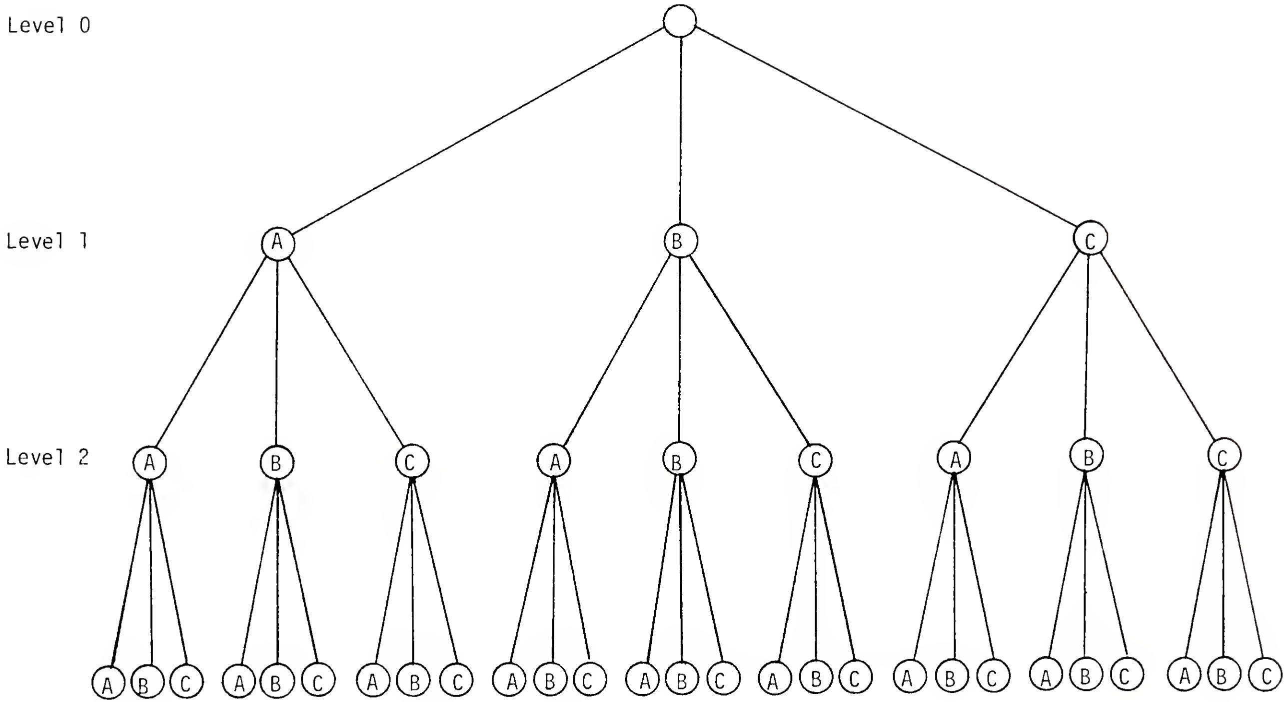

Figure 4.3. Search tree for enumeration algorithm.

In the enumeration algorithm (called TREE), if the search is in a given node in a given level, the search will not move to another node in that level until all the nodes connected to the present node and below it in the search tree have been evaluated. Consider, again, Figure 4.3. Each node can be identified by a series of letters determined by its position and those of its ancestors in their respective levels in the search tree. The first node in level1 is A. The second node in level 2 is AB, and the sixth node in level 3 is ABC. As an example of how the algorithm TREE works, if the search is in node B in level1, it has already finished with node A and has not started node C. It will not start node C until nodes BAA, BAB, BAC, BBA, BBB, BBC, BCA, BCB, and BCC (corresponding to complete schedules) have, in effect, been evaluated. Upper and lower bounds may pare down the tree or there may not be enough jobs in a product to justify lower level nodes, but that still has the effect of evaluating these nodes and not finding a better solution.

The enumeration algorithm TREE uses the separability presented in Property 4.1 to increase its efficiency. Each separable section is solved as an individual problem. All future discussion of the workings of TREE assumes it is dealing with a separable section.

Each of the separable sections is solved for its lower bound by the LOWBND subroutine. Property 4.2 shows that this lower bound supplies a lower bound for any partial schedule found by TREE.

Another property of the problem is used to eliminate the permutations of jobs within a period in the optimal solution. The immediate successor, if any exists, of job i is job i+1. Jobs from product p+1, if any, have a higher number than jobs from product p. Thus, there exists an optimal solution which has jobs in each period arranged on the machines

according to increasing job number. To take advantage of this property and eliminate the permutations of the jobs in each period, the enumeration algorithm only considers for machine j+1 in period t a job with a higher number than that of the job on machine j.

Consider the situation in which each period has two machines and Figure 4.3 is the search tree for only the first three levels. Levels 1 and 2 correspond to the machines in period 1. Level 3 is the first machine in period 2. Elimination of permutations of jobs within periods has the effect of changing Figure 4.3 into Figure 4.4.

In another attempt to pare down the tree in the enumeration search, pseudoprecedence constraints between jobs of different products are developed. In addition to its deferral cost, job i has a <u>pseudocost</u>. This cost equals the deferral cost of the "cheapest" job from among a set of jobs. The set of jobs contains only job i and those of its predecessors which fall into the same separable section as job i. Let jobs i1 and i2 be in the same separable section but from different products. Job i1 is a <u>pseudopredecessor</u> of job i2 if the pseudocost of job i1 is greater than the deferral cost of job i2, less than that of the immediate predecessor of job i2, and no less than the deferral cost of any successor of job i2 in the same separable section. If job i1 has all the other qualifications but has a pseudocost equal to the deferral cost of job i2, it is the pseudopredecessor of job i2, unless job i2 qualifies to be the pseudo-predecessor of job i1. In this latter case where both job i1 and job i2 qualify to be the pseudopredecessor, the job having the lower product number becomes the pseudopredecessor.

Consider job i2 scheduled in period t. If job i1 is scheduled later than t but is available in t, it is a violation of the

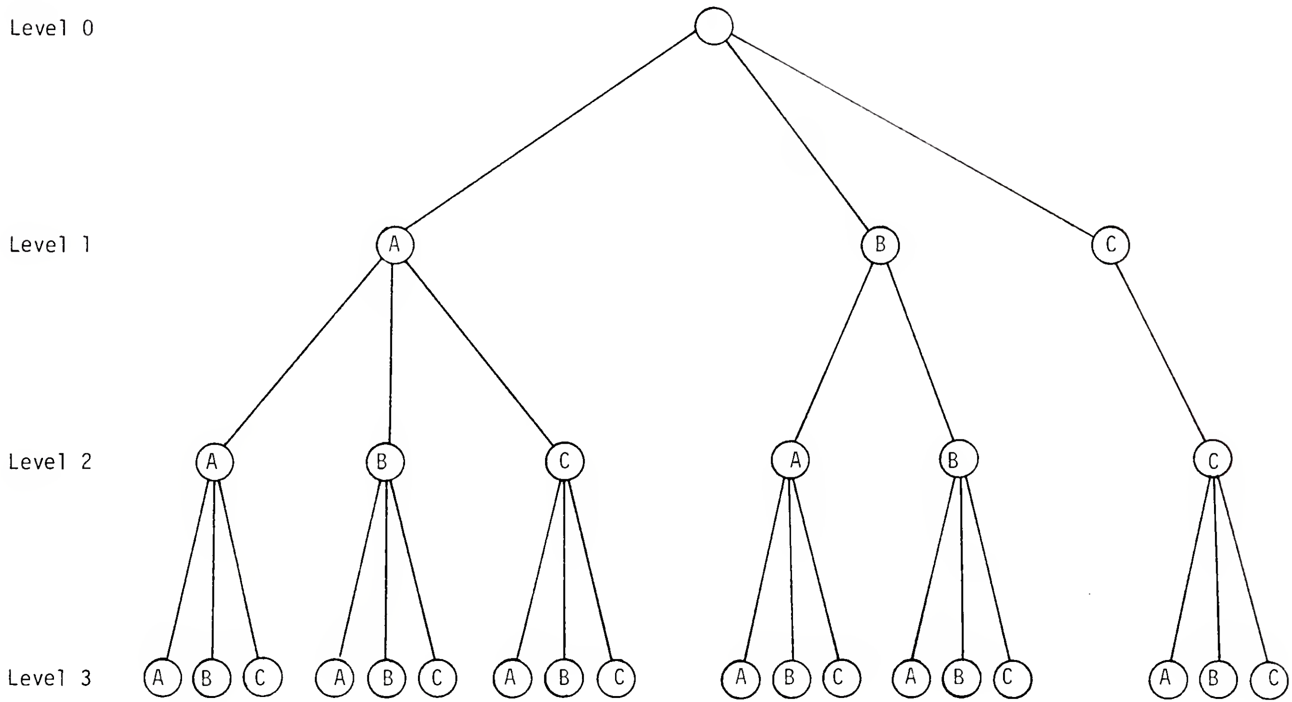

Figure 4.4. Search tree with permutations eliminated.

pseudoprecedence constraint. These contraints are maintained by the enumeration algorithm. It is easily shown, by a pairwise interchange proof, that there exists an optimal schedule in which the pseudo-precedence constraints are not violated.

Suppose the enumeration algorithm has just found a complete schedule. In addition, suppose that this complete schedule differs from the optimal schedule by the interchange of job i1 in period 1 and job i2 in period 2. At the point in the search at which job i1 has just been put in its optimal place and all the earlier levels agree with the optimal schedule, the search still has to work its way down to the bottom of the tree to complete the optimal schedule, even though the optimal assignment of the remaining jobs exists in the corresponding levels of that best complete solution found to date.

A property of the search technique used in this algorithm can be used to pare down the search tree. A complete schedule is not considered to have been found unless it is at least as good as the best previous schedule. It then becomes the new best previous schedule.

<u>Property 4.3</u>: Consider the situation in which a complete schedule has already been found by the algorithm and the present partial schedule, complete through exactly k levels, has the same jobs as the best previous schedule in its first k levels (though, not necessarily in the same order). For a scheduling horizon of K levels, the jobs in the last K-k levels of the best previous schedule are scheduled optimally within those levels and would optimally complete the present partial schedule.

<u>Proof</u>: Call level 1 the highest level in the search tree. In the search method of this algorithm, the job in level k is not changed

after finding the best previous schedule until all feasible orderings of the jobs in the lower levels have been tried. If any of those orderings were an improvement, it would result in an update of the best previous schedule. Thus, when the algorithm is about to change the job scheduled in level k, it has tested all the orderings of the jobs in the last K-k levels. The ordering in the last K-k levels of the best previous schedule must be optimal. If the algorithm comes to a situation in which the first k levels have the same jobs in both the present partial schedule and the best previous schedule, the candidates for the last K-k levels are those jobs in the last K-k levels of the best previous schedule. Since these are optimally ordered, they will optimally complete the present partial schedule.

Q.E.D.

This leads to a more powerful property. Let there be exactly k levels in the first t periods.

Property 4.4: Consider a present partial schedule which is completed only as far as level k1. If this schedule contains all the jobs from the first t periods and j jobs from period t+1 of the best previous schedule (where k1 = k+j) and if at least one job from period t is now in t+1, the present partial schedule can be completed optimally by scheduling the remaining jobs in the same periods which they occupy in the best previous schedule.

Proof: In order for a job to have moved from the first t periods of the best previous schedule to period t+1 of the present partial schedule, the algorithm must have backed up to at least level k before moving down the tree to fill the first j levels of period t+1 in the present partial schedule. Thus, by Property 4.3, the jobs in the last K-k levels of the

best previous schedule are scheduled optimally within those levels. Since the j jobs from level t+1 of the best previous schedule are from the same period, there exists a permutation of the best previous schedule, also optimal in the last K-k levels, in which those j jobs are in the first j levels of period t+1. The permutation has its last K-kl jobs scheduled optimally within those levels. The present partial schedule has the same jobs in the first kl levels as the permutation. By Property 4.3, the present partial schedule could be completed optimally by scheduling the remaining jobs in the same levels (thus, the same periods) as in the permutation.

Q.E.D.

The enumeration subroutine TREE has been developed in this section to take advantage of Property 4.3 and the other constraints, properties, and techniques mentioned above to pare down significantly the search tree. This paring makes practical the solution of a large number of randomly generated problems in order to test the accuracy and efficiency of the heuristics SCHED1 and SCHED2. The results are discussed in the next section.

4.5 Testing and Results

The previous sections have developed heuristic methods and the tools for testing their efficiency and accuracy. This section describes the testing procedure and the statistics used for evaluation. The results appear in tables. Those results of greatest interest have been expressed in graphs to demonstrate the effect of different levels of multiple interchanges.

To evaluate the accuracy of the heuristics a standard problem was developed to act as a basis for randomly generated test problems.

Five products were designated. The total number of jobs was varied as an input parameter. The number of jobs contributed by each product was decided randomly for each product, as were the availability times of each job. These times were in the interval from period 1 to 25. The overall scheduling horizon was 50 periods.

By way of explanation of the statistics presented, consider the example of three problems, A, B, and C. Let the values of their heuristic solutions be A1, B1, and C1, respectively. Their optimal values are A2, B2, and C2. Suppose the heuristic only solves C optimally; thus, C1 = C2. The heuristic has solved 66.7% of the problems nonoptimally.

The percent error for all problems in the experiment is

$$\left[\frac{A1 + B1 + C1}{A2 + B2 + C2} - 1\right] \times 100.$$

The percent error for nonoptimal problems in the experiment is

$$\left[\frac{A1 + B1}{A2 + B2} - 1\right] \times 100.$$

The average percent error for all problems is the error that can be expected in using the heuristic. It is calculated as

$$\left[\frac{\frac{A1}{A2} + \frac{B1}{B2} + \frac{C1}{C2}}{3} - 1\right] \times 100.$$

The average percent error for nonoptimal problems is the error realized when the heuristic does not find an optimal solution. It is calculated as $\left[\frac{\frac{A1}{A2} + \frac{B1}{B2}}{2} - 1\right] \times 100.$

In one set of problems, four identical machines were allocated to each period in the horizon. There were 115 randomly generated

problems, each having 50 jobs to be scheduled. The figures in Table 4.1 summarize the results of these tests.

Table 4.1 is partitioned into three blocks. The first two blocks represent the individual heuristics. The last block represents the joint heuristic created by using the better of the individual heuristic results from each problem. Within each block there are four columns. The heading on each column indicates the highest level of interchange used to augment the heuristic. For each level of interchange, all lower levels have been used in conjunction with it. The graphs in Figures 4.5, 4.6, and 4.7 illustrate some of the statistics from Table 4.1.

Table 4.2 is laid out the same as Table 4.1. It presents the same information for the 93 randomly generated problems having two identical machines allocated to each period.

With two exceptions all the measures of performance shown in Tables 4.1 and 4.2 improve as the level of interchange increases. One of the more interesting statistics, the percent of problems solved optimally, shows steady improvement (up to 98% optimal) with the level of interchange. (This is easily seen in Figures 4.5 through 4.10.) As with most of the statistics, this is especially true for the problems with more machines. The rate of improvement falls off for the two-machine problems after the pairwise interchange is implemented. This may be explained by SCHED2 without interchanges being an optimal algorithm for the single-machine problem having an availability time of zero for all jobs. The design of SCHED1 also seems to favor problems with a lower number of machines.

The first inconsistency in all this appears in the two percent-error-for-the-nonoptimal-problems statistics for the two-machine problems. They approach 1% for the four-machine problems compared to 9% for the two-machine

Table 4.1. Statistics from 115 problems having 4 machines and 50 jobs

Heuristic	SCHED 1				SCHED 2				JOINT			
Interchange	4 WAY	3 WAY	2 WAY	NONE	4 WAY	3 WAY	2 WAY	NONE	4 WAY	3 WAY	2 WAY	NONE
Percent of Problems Solved Optimally	93.0	66.1	55.7	0.0	96.5	85.2	79.1	33.9	98.3	88.7	85.2	33.9
Percent Error For All Problems	0.712	4.51	6.26	78.9	0.0697	0.737	1.066	6.82	0.025	0.458	0.642	6.78
Average Percent Error for All Problems	0.53	3.46	8.93	92.5	0.068	0.76	1.03	6.03	0.018	0.299	0.703	5.91
Percent Error For Nonoptimal Problems	5.77	9.64	10.99	78.9	1.42	3.51	5.21	9.02	0.762	2.68	3.33	8.96
Average Percent Error for Non-optimal Problems	7.60	10.2	20.1	92.5	1.96	5.17	4.96	9.12	1.01	2.64	4.76	8.95
Range of Percent Error for Non-optimal Problems	0.23- 19.4	0.23- 26.6	0.23- 73.6	5.66- 379.	0.23- 4.17	0.23- 34.9	0.47- 34.9	0.19- 42.9	0.23- 1.79	0.23- 4.69	0.23- 34.9	0.19- 42.9
Average Time For All Problems (10^{-2} SEC)	.8348	.6740	---	---	.7238	.6003	---	---	1.5586	1.2743	---	---

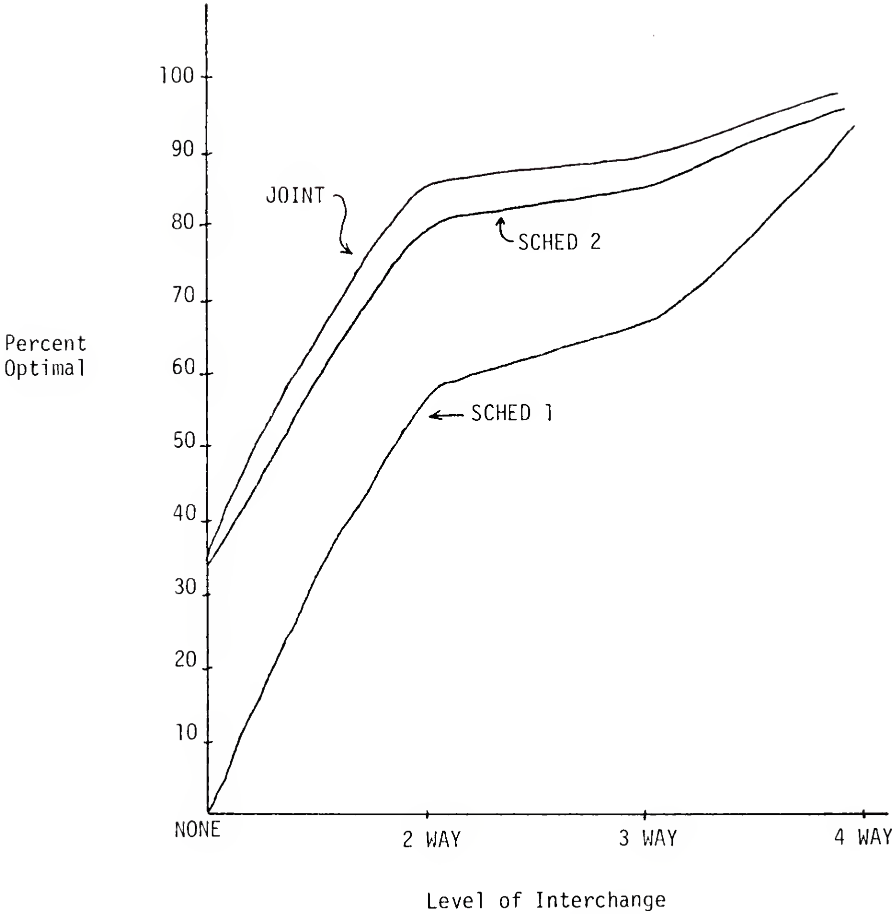

Figure 4.5. Graphs of level of multiple interchange versus percent of problems solved optimally for 4 machines and 50 jobs

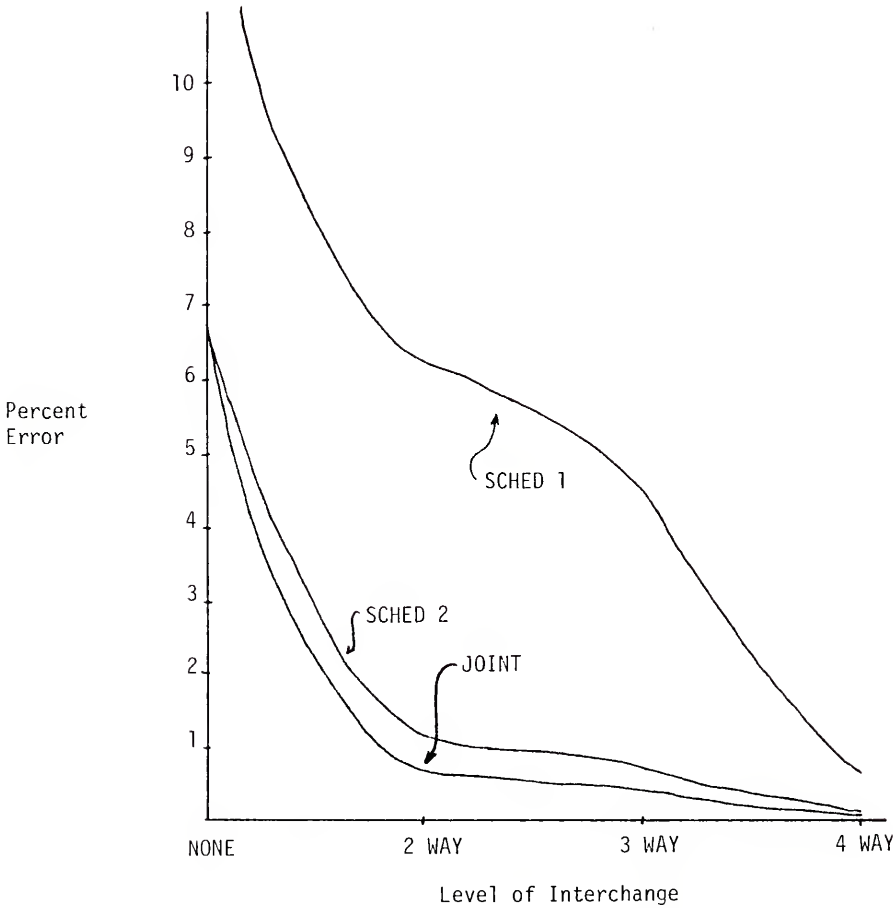

Figure 4.6. Graph of level of multiple interchange versus percent error for all problems for 4 machines and 50 jobs

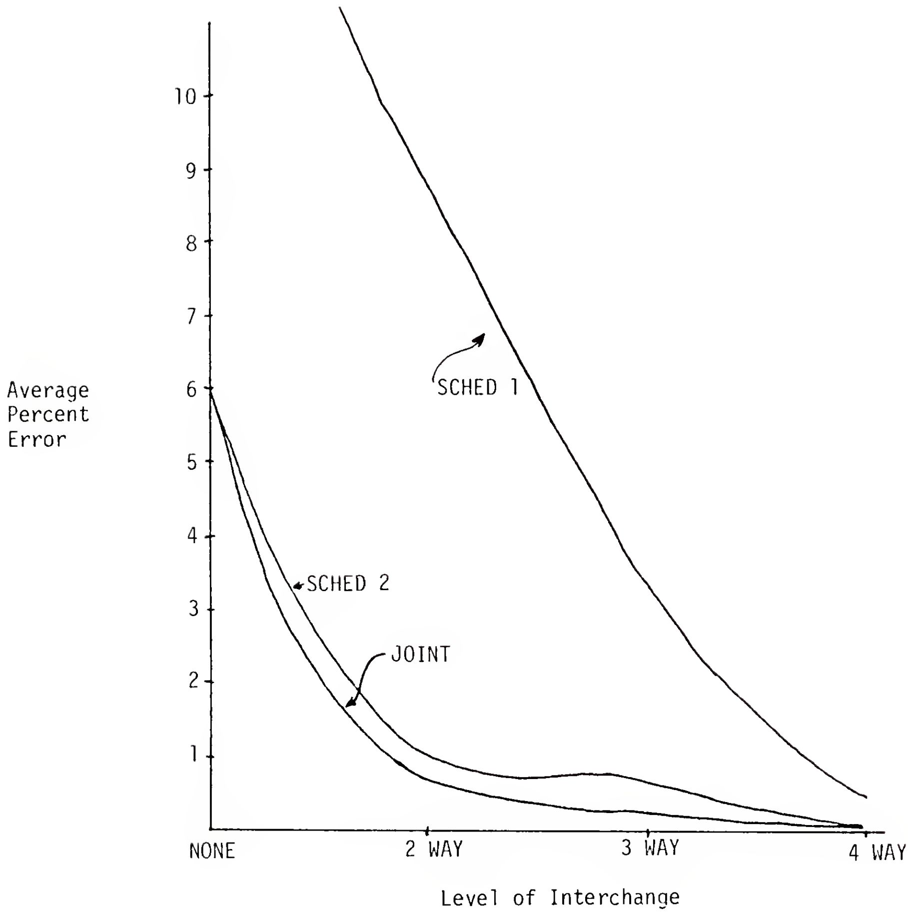

Figure 4.7. Graph of level of multiple interchange versus average percent error for all problems for 4 machines and 50 jobs

Table 4.2. Statistics from 93 problems having 2 machines and 30 jobs

Heuristic	SCHED 1				SCHED 2				JOINT			
Interchange	4 WAY	3 WAY	2 WAY	NONE	4 WAY	3 WAY	2 WAY	NONE	4 WAY	3 WAY	2 WAY	NONE
Percent of Problems Solved Optimally	96.8	89.3	69.9	0.0	97.9	94.6	85.0	41.9	98.9	98.9	93.6	41.9
Percent Error For All Problems	0.304	0.918	4.57	72.0	0.22	0.479	1.22	4.94	0.169	0.169	0.456	4.94
Average Percent Error For All Problems	0.18	0.75	3.86	83.7	0.14	0.41	1.199	5.44	0.097	0.097	0.310	5.44
Percent Error For Nonoptimal Problems	5.15	7.28	12.1	72.0	7.02	8.02	6.87	8.08	9.09	9.09	4.97	8.09
Average Percent Error for Non-Optimal Problems	5.57	7.01	12.8	83.7	6.53	7.61	7.96	9.37	9.01	9.01	4.80	9.37
Range of Percent Error for Non-optimal Problems	2.32-9.01	1.77-12.3	1.77-45.7	22.6-284.	4.04-9.01	3.45-11.97	1.09-31.3	0.8-32.2	9.01-9.01	9.01-9.01	1.1-11.7	0.88-32.2
Average Times For All Problems (10^{-2} SEC)	.56898	.49025	.38601	.1825	.52930	.45089	.29701	.2290	1.0983	.94114	.68302	.4115

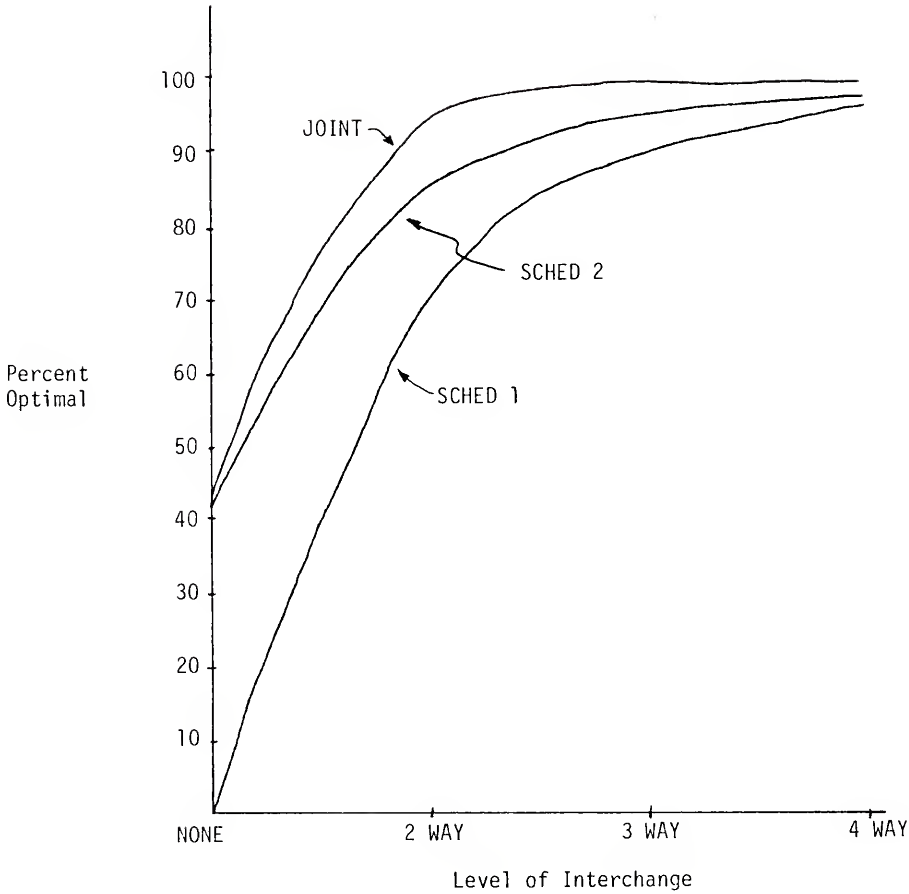

Figure 4.8. Graph of level of multiple interchange versus percent of problems solved optimally for 2 machines and 30 jobs

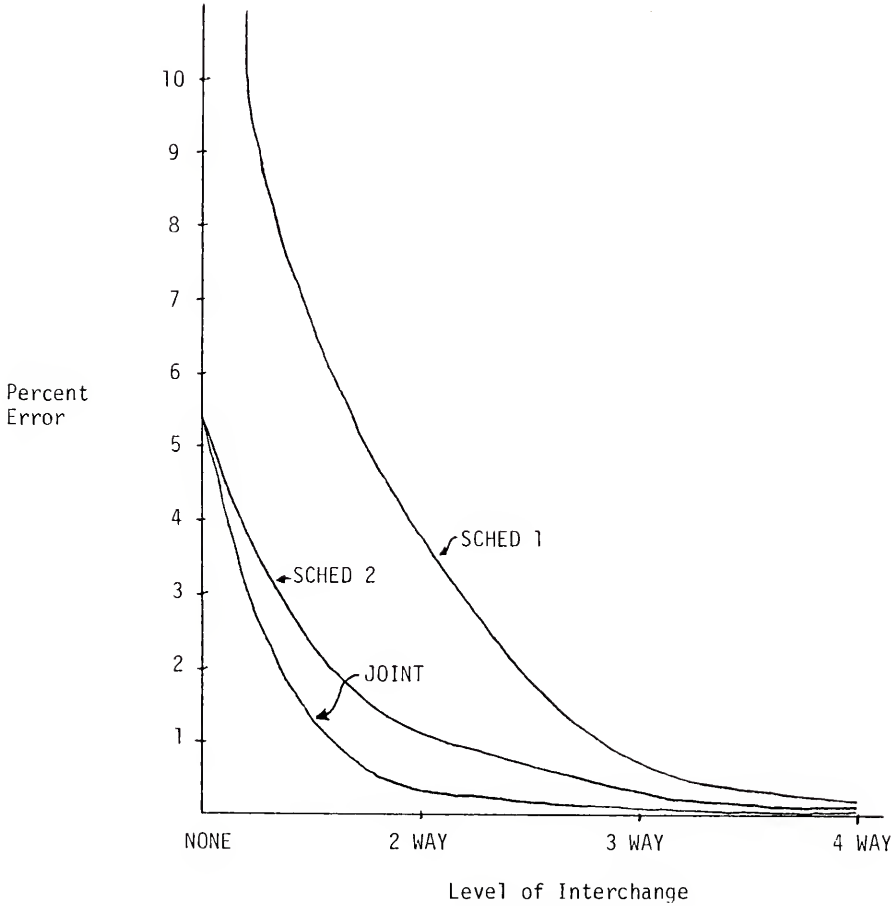

Figure 4.9. Graph of level of multiple interchange versus percent error for all problems for 2 machines and 30 jobs

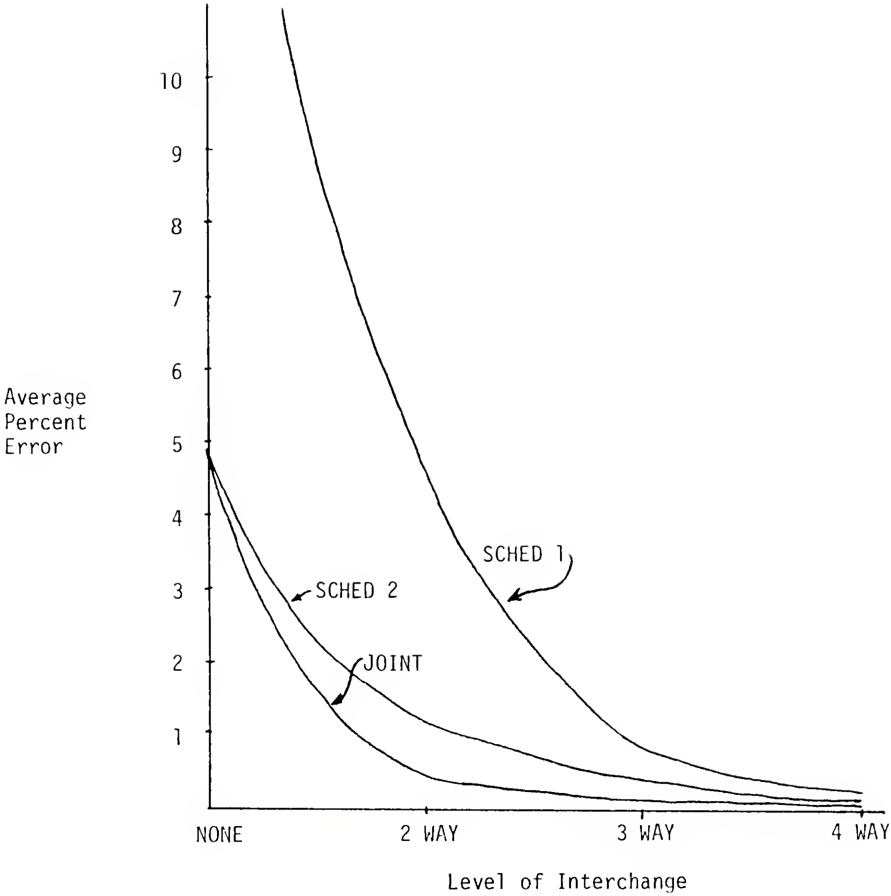

Figure 4.10. Graph of level of multiple interchange versus average percent error for all problems for 2 machines and 30 jobs

problems. Using the two-way interchange, the joint efforts of the two heuristics yielded six nonoptimal problems. They ranged in percent error from 1.1 to 11.7. The three-way interchange caused the joint effort to solve optimally all but one problem. Unfortunately, that problem had the second worst percent error and did not improve. This caused an increase in the percent error and average percent error statistics for nonoptimal problems. The four-way interchange had no effect on the joint effort. Similar situations caused the inconsistency in the percent error for non-optimal problems for SCHED2 for two machines and in the average percent error for nonoptimal problems for SCHED2 for four machines.

Complete time statistics are only available for the two-machine problems. However, the apparent linear relationship between the two and four-machine problems implies the same conclusions can be drawn for both sets of problems. The running time appears to be a concave increasing function of the level of interchange, as is the percent of problems solved optimally (Figures 4.5, 4.8, and 4.11).

In this section it was explained how the problems were randomly generated to test the accuracy and efficiency of the heuristics and the multiple interchange methods which augment them. The statistics were compiled by comparing the results from over 200 randomly generated problems with the optimal solutions found by an enumeration routine. Tables 4.1 and 4.2 showed a steady increase in accuracy in the individual heuristics--up to as much as 98% of the problems solved optimally--as the level of interchange increased. The augmentation of the heuristics was accomplished while maintaining execution time within three times that of the unassisted heuristic in the worst case.

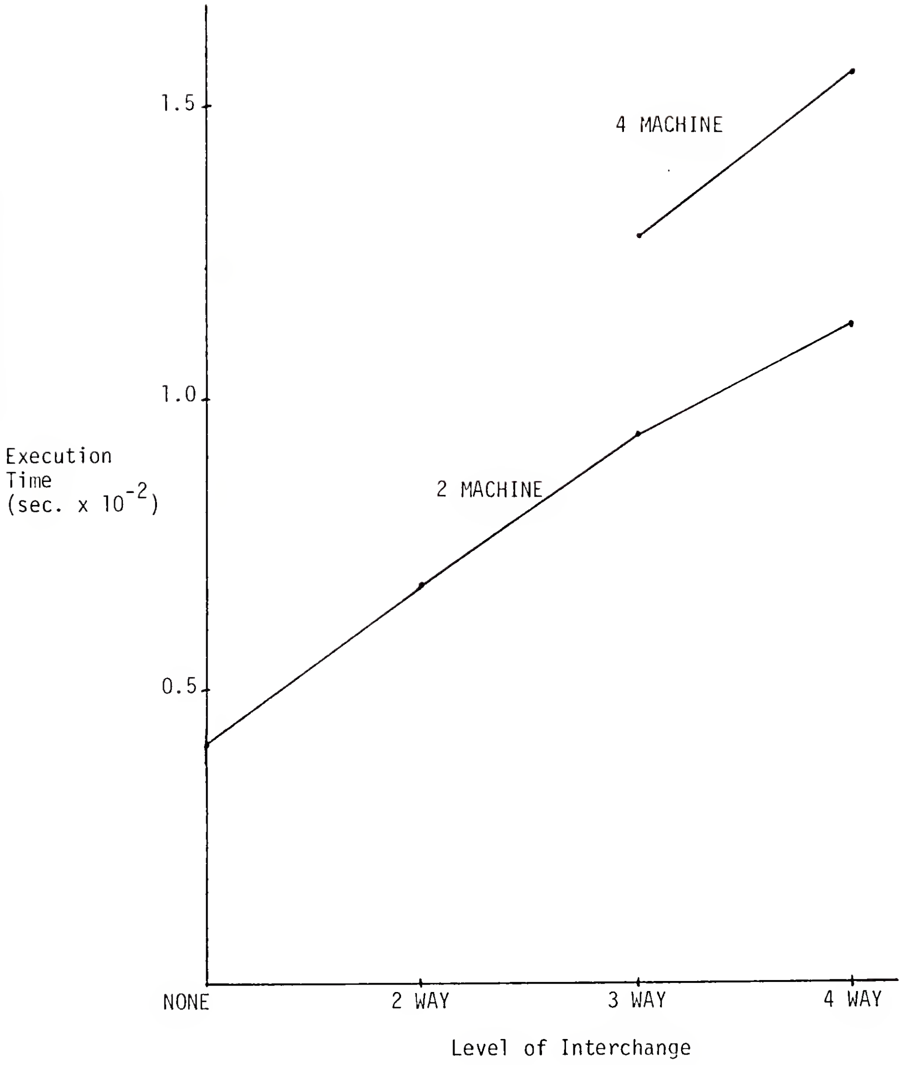

Figure 4.11. Graph of joint heuristic execution time versus level of interchange.

4.6 Conclusions

The multi-machine, multi-product scheduling problem whose jobs have serial precedence constraints among them and availability times can be formulated as a transportation problem with complicating constraints. Since this is such a difficult problem to solve, a pair of heuristics were developed along with a multiple interchange method to augment them. An enumeration routine checked their accuracy on over 200 randomly generated problems.

The key factor in the efficiency of the enumeration algorithm is the number of levels in the search tree for a separable problem section. The limit seems to be about 30 levels before the execution time explodes. This limits the ability for testing problems larger than tested here.

The number of lines of Fortran code grows from 50 to 140 to 312 for the two-way, three-way, and four-way interchange routines, respectively. In each routine approximately 25% of these are conditional statements. While the execution time appears to be no worse than a linearly increasing function of the level of interchange, the amount of code appears to be a convex increasing function. This implies that a design trade-off between code development and solution improvement exists instead of an expected trade-off between execution time and solution improvement. In the worst case, 50% of the nonoptimal problems are solved optimally by increasing the level of interchange by one for an individual heuristic. The use of multiple interchanges appears to be a valuable addition to any heuristic for this problem.

This success of the joint use of the heuristics SCHED1 and SCHED2 indicates that it would be a good practice to use more than one

heuristic on the problem. Combined with the multiple interchange method, 98.5% of the problems were solved optimally.

CHAPTER 5

SUMMARY AND SUGGESTIONS FOR FUTURE RESEARCH

In this dissertation a multi-stage, multi-machine, multi-product production scheduling problem was considered. First, it was formulated as an integer program. From this initial problem two cases were investigated--a frontal production bottleneck and a posterior production bottleneck. An algorithm which produces an optimal schedule was developed for the first case. Two heuristics (and an effective multiple interchange method to augment them) were developed for the other case.

The frontal production bottleneck is created by the production rate for each product being a nondecreasing function of the level of the stage within the production system ($p_i^j \leq p_i^{j1}$ for $j \in S(j1)$). It was shown that with reasonable conditions placed on the costs, the number of machines per stage (an upper bound), and the accuracy of the short term demand forecast and after a start-up period a single pass algorithm can be used to solve the problem optimally. Counterexamples were provided to demonstrate that the algorithm can give suboptimal solutions when any of these conditions are not met.

Attempts were made, unsuccessfully, to determine bounds on the error associated with the start-up period and damaging fluctuations in the short term demand forecast. Since these bounds would be very valuable in the use of the solution procedure, their further analysis is a desirable subject for continuing and future research.

The restrictions on the number of machines per stage and on the costs in the frontal bottleneck problem are another area for future study, which might well focus on the development of methods different from the one presented here.

The posterior bottleneck problem is caused by the production rate for each product being a nonincreasing function of the level of the stage within the production system ($p_i^j \geq p_i^{j1}$ for $j \epsilon S(j1)$). The only restriction placed on the problem in this case is a lower bound on the number of machines in each stage. It was shown that this multi-stage problem can be reformulated as a single-stage scheduling problem in which the jobs have availability times and are partially ordered by serial precedence constraints among them. Two heuristics for the problem were presented. An efficient multiple interchange method was developed, and when used to augment the heuristics, resulted in a significant increase in the percentage of problems solved optimally. Combining the two heuristics and using the highest available level of interchange resulted in the optimal solution of over 98% of the randomly generated test problems.

It appears that there exists a tradeoff point between developmental cost and increased heuristic accuracy. It would be interesting to determine that relationship, but in a more general scheduling context.

Investigating the effect of lowering or removing the lower bounds on the number of machines in each stage of the posterior bottleneck would be a difficult project since supplier jobs could not always immediately precede their consumers. The problem could not be reformulated as a single-stage problem.

As far as the original multi-stage problem is concerned, there are many questions remaining to be answered for the cases in which the

production rates are not as nicely behaved as in the cases discussed here. What happens if one or more products experience a reversal in the change of their production rates as their level in the production system increases? Does the reversal make the problem too complex to develop optimality conditions? Would a less complex network of stages open the door for a new solution method?

These are a few of the avenues for future research. It has been shown that the method of Chapter 2 does not extend to any of these cases. Therefore, they probably require a completely different approach.

APPENDIX

COMPUTER PROGRAMS

```
C*************************************************************************
C*     MAIN PROGRAM                                                      **
C*************************************************************************
C*                                                                       **
C*    IT IS ASSUMED THAT EACH TIME PERIOD HAS AT LEAST ONE MACHINE.      **
C*    THE JOBS FOR EACH PRODUCT ARE ARRANGED IN STRINGS BY PRECEDENCE    **
C*    CONSTRAINTS WHICH ALLOW A JOB TO BE SCHEDULED NO EARLIER THAN      **
C*    ITS PREDECESSOR. THE PREDECESSOR OF THE EARLIEST JOB AND THE       **
C*    SUCCESSOR OF THE LATEST JOB FOR EACH PRODUCT IS ZERO. EACH JOB     **
C*    HAS AN AVAILABILITY DATE AND POSITIVE INTEGER DEFERRAL COST AND    **
C*    PROCESSING TIME = 1 PERIOD.                                        **
C*                                                                       **
C*************************************************************************
C*                                                                       **
C*    IMPORTANT MATRICES AND VARIABLES                                   **
C*                                                                       **
C*    JMACH = WORKING ARRAY CONTAINING NUMBER OF MACHINES REMAINING      **
C*               TO BE SCHEDULED IN EACH PERIOD                          **
C*    JOB    = ARRAY CONTAINING INFORMATION ON EACH JOB                  **
C*               EACH ROW IS A JOB                                       **
C*               COLUMNS:      JOB                                       **
C*                           1 IS THE PRODUCT NUMBER;                    **
C*                           2 IS THE AVAILABILITY DATE;                 **
C*                           3 IS THE PREDECESSOR;       ;               **
C*                           4 IS THE SUCCESSOR.                         **
C*    JPROD  = ARRAY CONTAINING THE NUMBER OF THE EARLIEST UNSCHED-      **
C*               ULED JOB OF EACH PRODUCT IN COLUMN 1 AND THE NUMBER     **
C*               OF THE NEXT UNFORCED SUCCESSOR OF THE LAST AVAILABLE    **
C*               JOB ON THE STRING IN COLUMN 2                           **
C*    JFPROD = ARRAY CONTAINING THE NUMBER OF THE FIRST JOB OF EACH      **
C*               PRODUCT                                                 **
C*    LBND  = ARRAY WHICH AT FIRST CONTAINS THE JOB SCHEDULE FROM THE    **
C*             GREEDY (LOWER BOUND) ALGORITHM. LATER IT CONTAINS, IN THE **
C*             ITH ENTRY, THE COST OF THE GREEDY SCHEDULE FROM LEVEL I   **
C*             FORWARD.                                                  **
C*    MACH  = ARRAY CONTAINING THE NUMBER OF MACHINES IN EACH PERIOD     **
C*    MSCHED = THE NUMBER OF MACHINES SCHED IS DIMENSIONED TO HAVE FOR   **
C*               EACH PERIOD                                             **
C*    NJOB   = NO. OF JOBS TO BE SCHEDULED                               **
C*    NMACH = WORKING ARRAY CONTAINING NUMBER OF MACHINES REMAINING      **
C*              TO BE SCHEDULED IN EACH PERIOD                           **
C*    NPROD  = NUMBER OF PRODUCTS                                        **
C*    RJOB   = REAL ARRAY CONTAINING DEFERRAL COSTS FOR EACH JOB         **
C*    SCHED = SCHEDULE. FIRST SUBSCRIPT IS THE NUMBER OF THE MACHINE     **
C*              IN THE PERIOD.  THE SECOND SUBSCRIPT IS THE PERIOD NUM-  **
C*              BER.  THE THIRD SUBSCRIPT IS THE SCHEDULING METHOD.      **
C*    SJOB   = ARRAY WHOSE ITH ENTRY IS 1 IF JOB I IS SCHEDULED IN THE   **
C***            GREEDY SCHEDULE                                          **
C*    TIME  = LENGTH OF SCHEDULING HORIZON                               **
C*    MAC    = NUMBER OF MACHINES ACTIVE IN EACH PERIOD                  *
C*    PROB2  = NUMBER OF PROBLEMS TO BE SOLVED IN THE RUN                *
C*    BUSS   = NUMBER OF SECONDS TREE IS ALLOWED TO WORK ON A PROBLEM    *
C*               BEFORE INTERRUPTION                                     *
C*    ALIMIT = FRACTION OF HORIZON OVER WHICH AVAILABILITY TIMES CAN     *
C*               OCCUR                                                   *
C*    SEED   = INITIAL SEED FOR RANDOM NUMBER GENERATOR                  *
C*    GEN    = LOGICAL VARIABLE WHICH, WHEN FALSE, BYPASSES RANDOM       *
C*               GENERATION OF PROBLEMS AND ALLOWS INPUT OF SPECIFIC     *
C*               PROBLEM.                                                *
C*    PRINT  = LOGICAL VARIABLE WHICH, WHEN FALSE, BYPASSES THE          *
C*               PRINTING OF THE HEURISTIC SCHEDULES AND PRODUCT         *
C*               PRECEDENCE CONSTRAINTS                                  *
C*    BUZ    = LOGICAL VARIABLE WHICH, WHEN FALSE, STOPS EXECUTION       *
C*               OF TREE AFTER 'BUSS' SECONDS                            *
C*    ADRIAN = LOGICAL VARIABLE WHICH, WHEN TRUE, PREVENTS EXECUTION     *
C*               OF TREE                                                 *
C*   IFLAG   = LOGICAL VARIABLE WHICH, WHEN FALSE, PREVENTS EXECUTION    *
```

```
C*              OF PAIR.CHANGE AND WAY4                                   *
C*    FLAG1   = LOGICAL VARIABLE WHICH, WHEN FALSE, PREVENTS EXECUTION   *
C*              OF CHANGE AND WAY4                                        *
C*    FLAG2   = LOGICAL VARIABLE WHICH, WHEN FALSE, PREVENTS EXECUTION   *
C*              OF WAY4                                                   *
C*                                                                        *
C*************************************************************************
      IMPLICIT INTEGER*4 (A-Z)
      REAL Z,TTIME,ALIMIT, RUSS
      LOGICAL  GEN,PRINT,BJZ,ADRIAN,FLAG,FLAG1,FLAG2
      COMMON LBND(1000),TLBND(1000), RPROD(20),JPROD(20),JPROD2(20),
     *         MACH(200),JMACH(200),NMACH(200),RJOB(200),SJOB(200),
     *         JOB1(200),JOB2(200),JOB3(200),JOB4(200),DJOB(200),
     *         SCHED(5,200,3),TIME,NJOB,NPROD,HCOST,COPT,MSCHED,TIME1,
     *         TIME2,PRINT
C*************************************************************************
C*    INPUT PROBLEM DATA AND PRINT OUT PROBLEM CHARACTERISTICS            *
C*************************************************************************
      CALL TSKIME
      READ(5,4) NCARD
4     FORMAT(I3)
      WRITE (6,5)
5     FORMAT ('1')
      CARD=0
9     READ (5,10) NJOB, NPROD, TIME, MAC, MSCHED, PROB2, RUSS, ALIMIT,
     *            SEED, GEN, PRINT, BJZ, ADRIAN, FLAG, FLAG1, FLAG2
      PROB1 = 0
10    FORMAT (6I5, 2F10.5, I20, 7L1)
11    WRITE (6,12) NJOB, NPROD, TIME, MAC
12    FORMAT (///, ' THE NUMBER OF JOBS THAT ARE SCHEDULED IS    ', I5,
     *        //, ' THE NUMBER OF PRODUCTS IS   ', I5,
     *        //, ' THE NUMBER OF TIME PERIODS IS ', I5,
     *        //, ' THE NUMBER OF MACHINES AVAILABLE PER PERIOD IS', I5)
      DO 15 I = 1, TIME
      MACH(I) = MAC
15    CONTINUE
      IF (GEN) GO TO 28
      WRITE (6,20)
20    FORMAT (/////, ' JOB NUMBER', 14X, 'JOB MATRIX', 13X,
     *        'RJOB VECTOR', ///)
      DO 25 I = 1, NJOB
      READ (5,23) J, JOB1(J), JOB2(J), JOB3(J), JOB4(J), RJOB(J)
23    FORMAT (6I5)
      WRITE (6,26) J, JOB1(J), JOB2(J), JOB3(J), JOB4(J), RJOB(J)
25    CONTINUE
26    FORMAT (I5, 4I10, 2X, I5)
      GO TO 31
28    CALL GENER(SEED, ALIMIT)
C*************************************************************************
C*    INITIALIZE PROGRAM MATRICES. SCHED MUST BE INITIALIZED TO ITS      *
C*    FULL DIMENSIONS.                                                    *
C*************************************************************************
31    DO 33 J = 1, NJOB
      IF (JOB3(J) .NE. 0) GO TO 33
      JPROD(JOB1(J) ) = J
33    CONTINUE
      DO 35 K = 1, 3
      DO 35 I = 1, MSCHED
      DO 35 J = 1, 200
      SCHED(I,J,K) = 0
35    CONTINUE
C*************************************************************************
C*    DETERMINE THE JOB SCHEDULE VIA THE SCHED1 ALGORITHM AND PRINT OUT*
C*    THE RESULTING JOB SCHEDULE AND STATISTICS.                          *
C*************************************************************************
      HCOST = 1E6
      Z = TTIME(2)
      CALL SCHED1(FLAG,FLAG1,FLAG2)
      Z = TTIME(2)
      COST1 = COPT
      IF ( .NOT. PRINT) GO TO 72
      WRITE (6,50)
```

```
60    FORMAT (/////, ' THE JOB SCHEDULE UNDER THE SCHED1 METHOD IS')
      WRITE (6,65)
65    FORMAT (///, ' TIME PERIOD', 20X, 'MACHINE')
      WRITE (6,66) (I, I = 1, MSCHED)
66    FORMAT (/, 25X, 14I5)
      WRITE (6,67)
67    FORMAT (//)
      DO 70 J = 1, TIME
      LIMIT = MACH(J)
      WRITE (6,71) J, (SCHED(I, J, 1), I = 1, LIMIT)
70    CONTINUE
71    FORMAT (5X, I5, 15X, 14I5)
72    WRITE (6,73) COST1
73    FORMAT (//, ' THE COST OF PRODUCTION UNDER THE SCHED1 PRODUCTION',
     *       ' SCHEDULE IS', I10)
      WRITE (6,74) Z
74    FORMAT (//, ' THE EXECUTION TIME REQUIRED FOR THIS METHOD WAS',
     *       2X, E10.5)
C*****************************************************************************
C*    DETERMINE THE JOB SCHEDULE VIA THE SCHED2 ALGORITHM AND PRINT OUT*
C*    THE RESULTING JOB SCHEDULE AND STATISTICS.                        *
C*****************************************************************************
      Z = TTIME(2)
      CALL SCHED2(FLAG,FLAG1,FLAG2)
      Z = TTIME(2)
      COST2 = COPT
      IF (.NOT. PRINT) GO TO 61
      WRITE (6,75)
75    FORMAT (/////, ' THE JOB SCHEDULE UNDER THE SCHED2 METHOD IS')
      WRITE (6,65)
      WRITE (6,66) (I, I = 1, MSCHED)
      WRITE (6,67)
      DO 60 J = 1, TIME
      LIMIT = MACH(J)
      WRITE (6,71) J, (SCHED(I, J, 2), I = 1, LIMIT)
60    CONTINUE
61    WRITE (6,63) COST2
63    FORMAT (//, ' THE COST OF PRODUCTION UNDER THE SCHED2 SCHEDULE ',
     *       'IS', I10)
      WRITE (6,74) Z
      IF(ADRIAN) GO TO 200
C*****************************************************************************
C*    DETERMINE THE UPPER BOUND SOLUTION BY SOLVING A RELAXED PROBLEM, *
C*    THEN PRINT OUT THE RESULTING JOB SCHEDULE AND STATISTICS.        *
C*****************************************************************************
      Z = TTIME(2)
      CALL LOWBND
      Z = TTIME(2)
      WRITE (6,64) Z
64    FORMAT (//, ' THE EXECUTION TIME REQUIRED FOR THE LOWBND',
     *       ' CALCULATIONS WAS', 2X, E10.5)
C*****************************************************************************
C*    SOLVE THE SCHEDULING PROBLEM VIA THE TREE BRANCH AND BOUND ALGOR-*
C*    ITHM AND PRINT OUT THE JOB SCHEDULE AND STATISTICS.               *
C*****************************************************************************
      Z = TTIME(2)
      CALL TREE(RUSS,BJZ)
      Z = TTIME(2)
      IF(BJZ) GO TO 85
      IF(Z .GT. RUSS) GO TO 200
85    WRITE(6,86)
86    FORMAT (/////, ' THE JOB SCHEDULE UNDER THE TREE METHOD IS' )
      WRITE (6,65)
      WRITE (6,66) (I, I = 1, MSCHED)
      WRITE (6,67)
      DO 90 J = 1, TIME
      LIMIT = MACH(J)
      WRITE (6,71) J, (SCHED(I,J,3), I = 1, LIMIT)
90    CONTINUE
      WRITE (6,95) COPT
95    FORMAT (//, ' THE COST OF PRODUCTION UNDER THE TREE SCHEDULE IS',
     *       I10)

      WRITE (6,74) Z
      IF (COPT .LT. HCOST) GO TO 140
      BEST = 1
      IF (COST2 .EQ. HCOST) BEST = 2
      WRITE (6,120)  BEST
120   FORMAT (/////, ' THE JOB SCHEDULE CONSTRUCTED BY SCHED ', I1,
     *       ' IS OPTIMAL')
140   WRITE(6,195)
195   FORMAT (///////)
200   IF (.NOT. GEN) GO TO 250
      WRITE (6,5)
      PROB1 = PROB1 + 1
      IF (PROB1 .LT. PROB2) GO TO 11
      CARD=CARD+1
      IF(CARD.LE.NCARD) GO TO 9
250   STOP
      END
```

```
C***********************************************************************
      SUBROUTINE TREE(RP,BJZ)
C***********************************************************************
C*                                                                     *
C*    SUBROUTINE TREE                                                  *
C*    A BOUNDED ENUMERATION METHOD FOR FINDING THE OPTIMAL SOLUTION    *
C*    WITH TIME DEPENDENT PRECEDENT CONSTRAINTS TO RESTRICT THE NUMBER *
C*    OF FEASIBLE SOLUTIONS THAT MUST BE EXAMINED.                     *
C*                                                                     *
C*    THE HEURISTICS WILL HAVE THE SAME NUMBER OF EMPTY MACHINES IN    *
C*    EACH PERIOD AS THE OPTIMAL SOLUTIONS HAVE.                       *
C*                                                                     *
C***********************************************************************
C*                                                                    **
C*    DISKD  = IS AN ARRAY USED FOR TRANSFERRING SCHEDULES INTO SCHED **
C*             AND REFERRING TO SCHED IN TERMS OF LEVELS              **
C*    DSCHED  = A WORKING ARRAY CONTAINING THE DUMMY SCHEDULE         **
C*    PLEV    = AN ARRAY OF POINTERS WHICH INDICATE WHICH PRODUCT IS  **
C*             UNDER CONSIDERATION ON EACH LEVEL                      **
C*                                                                    **
C*    IF LEVELS 1-(K-1) ARE SCHEDULED AND WE ARE LOOKING AT LEVEL K   **
C*    AND AT LEAST ONE SCHED WITH THE SAME SCHED IN 1 THRU (K-1) HAS  **
C*    BEEN COMPLETED OR STOPPED THEN ANY COMPLETE SCHED WHICH IS THE  **
C*    SAME IN 1 THRU (K-1) AND EMPTY IN K CAN BE NO BETTER THAN       **
C*    EXISTING SCHEDULES.                                             **
C*                                                                    **
C*    JMACH = AN ARRAY CONTAINING THE CUMULATIVE NUMBER OF MACHINES   **
C*            IN THE FIRST T PERIOD                                   **
C*                                                                    **
C***********************************************************************
C***********************************************************************
      IMPLICIT INTEGER*4 (A-Z)
      COMMON IEND(1000),TEND(1000),J1PROD(20),JPROD1(20),JPROD2(20),
     *          MACH(200),JMACH(200),NMACH(200),RJOB(200),SJOB(200),
     *          JOB1(200),JOB2(200),JOB3(200),JOB4(200),DJOB(200),
     *          SCHED(5,200,3),TIME,NJOB,NPROD,KCOST,COPT,NSCHED,TIME1,
     *          TIME2,PRINT
      REAL Z, TTIME, RUSS, BJ, RP
      LOGICAL PRINT,BJZ
      DIMENSION DISKD(1000,3),DSCHED(500),PLEV(500),J2PROD(20),PJOB(200)
      DIMENSION BJOB(200),MAXL(1000),ELEV(1000),B1(200),
     *             NEST(200),MAXT(200),C(200),HPRED(100,40),LPRED(100,40)
      EQUIVALENCE (DISKD(1,1), SCHED(1,1,1))
      CALL TSKTME
      RUSS = RP
      COPT = 0
      NLEV = 0
C***********************************************************************
C*    INITIALIZE IMPORTANT VECTORS                                     *
C***********************************************************************
      DO 10 I = 1,NJOB
      PJOB(I) = RJOB(I) * JOB2(I)
10    CONTINUE
      DO 25 I = 1, TIME
      NLEV = NLEV+NSCHED
      JMACH(I) = NLEV
25    CONTINUE
      DO 30 I = 1, NLEV
      PLEV(I) = 0
      DSCHED(I) = 0
30    CONTINUE
      DO 35 I = 1, NPROD
      JPROD1(I) = J1PROD(I)
35    CONTINUE
      DO 50 I = 1, NJOB
      HPRED(I,1) = 1
      LPRED(I,1) = 1
```

```
50      CONTINUE
        DO 55 J = 1, NJOB
        SJOB(J) = 0
55      CONTINUE
60      FORMAT (/////, ' THE SET OF TIME DEPENDANT JOB RELATIONSHIPS IS')
C*************************************************************************
C*      DECOMPOSE THE TIME HORIZON INTO SECTIONS WITH NO UNUSED MACHINES *
C*      IN ANY SECTION.                                                  *
C*************************************************************************
        TIME1 = 1
65      LEV1 = JMACH(TIME1) - MSCHED + 1
        DO 70 J = TIME1, TIME
        IF (SCHED (MACH(J), J, 1) .EQ. 0) GO TO 75
70      CONTINUE
        TIME2 = TIME
        GO TO 85
75      IF (SCHED(1,J,1) .EQ. 0) GO TO 80
        TIME2 = J
        GO TO 85
80      TIME2 = J - 1
C*************************************************************************
C*      LABEL THE LAST LEVEL IN THIS SEPARABLE PROBLEM SECTION WITH A JOB*
C*      AS 'LEV2'.                                                       *
C*************************************************************************
85      L2 = JMACH(TIME2) - MSCHED + MACH(TIME2)
        L1 = JMACH(TIME2) - MSCHED + 1
        DO 87 L = LEV1, L2
        BLEV(L) = 0
        MAXL(L)=0
87      CONTINUE
        DO 88 I = 1, NJOB
        BJOB(I) = BLEV + 1
        BT(I) = TIME2+1
88      CONTINUE
        DO 90 L = L1, L2
        IF (DISKD(L,1) .EQ. 0) GO TO 95
90      CONTINUE
        LEV2 = L2
        GO TO 97
95      LEV2 = L - 1
C*************************************************************************
C*      RECORD THE BEST HEURISTIC COST FOR THE TIME SECTION LEV1 - LEV2. *
C*************************************************************************
97      COST1 = 0
        NBS(TIME2+1) = 0
        COST = 0
        COST2 = 0
        HNUM = 2
        IF (TIME1 .EQ. TIME2) GO TO 105
        T2 = TIME2 - 1
        NBS(TIME2)=0
        MAXT(TIME2)=0
        DO 100 J = TIME1, T2
        MACHJ = MACH(J)
        NBS(J)=0
        MAXT(J)=0
        DO 100 I = 1, MACHJ
        COST1 = COST1 + RJOB( SCHED(I,J,1) ) * J
        COST2 = COST2 + RJOB( SCHED(I,J,2) ) * J
100     CONTINUE
105     DO 107 L = L1, LEV2
        COST1 = COST1 + RJOB( DISKD(L,1) ) * TIME2
        COST2 = COST2 + RJOB( DISKD(L,2) ) * TIME2
107     CONTINUE
        IF (COST1 .GT. COST2) GO TO 110
        COST2 = COST1
        HNUM = 1
110     COST1 = 0
        COST = COST2
        IF (TIME1 .EQ. TIME2) GO TO 434
        IF (COST2 .EQ. LBND(LEV1) ) GO TO 434
C*************************************************************************
```

```
C*      DETERMINE THE LAST JOB IN EACH PRODUCT STRING WHICH FALLS WITHIN *
C*      THIS SEPARABLE PROBLEM SECTION.                                  *
C************************************************************************
        DO 145 P = 1, NPROD
        J = JPROD1(P)
        J2PROD(P) = 0
        IF (J .EQ. 0) GO TO 145
        IF (JOB2(J) .GT. TIME2) GO TO 145
139     J2PROD(P) = J
140     J = JOB4(J)
        IF (J .EQ. 0) GO TO 145
        IF (JOB2(J) .GT. TIME2) GO TO 145
        GO TO 139
145     CONTINUE
C************************************************************************
C*      IF JOB I AND ALL OF ITS PREDECESSORS HAVE A DEFERRAL COST .GE.   *
C*      THAT OF JOB N, JOB I MAY BE A PSEUDO PREDECESSOR OF JOB N. EXA-  *
C*      MINE EACH PRODUCT STRING FOR SUCH CANDIDATE PREDECESSOR JOBS.    *
C************************************************************************
        DO 200 P = 1, NPROD
        I = JPROD1(P)
        IF (I .EQ. 0) GO TO 200
        LCOST= RJOB(I)
185     IF (RJOB(I) .GE. LCOST) GO TO 187
        LCOST = RJOB(I)
187     PJOB(I) = LCOST
        I = JOB4(I)
        IF (I .GT. 0) GO TO 185
200     CONTINUE
C************************************************************************
C*      IF A JOB N AND ALL OF ITS SUCCESSOR JOBS HAVE DEFERRAL COSTS     *
C*      WHICH ARE NOT .GT. JOB I, JOB I IS A PSEUDO PREDECESSOR OF JOB N.*
C************************************************************************
        DO 232 I = 1, NJOB
        IF ( SJOB(I) .GT. 0) GO TO 232
        IF ( JOB2(I) .GT. TIME2) GO TO 232
        RJI = PJOB(I)
        IVAL1 = JOB2(I)
        TYPE = JOB1(I) - 1
        IF (TYPE .EQ. 0) GO TO 216
C************************************************************************
C*      EXAMINE LOWER NUMBER PRODUCT STRINGS FOR PSEUDO SUCCESSOR JOBS TO*
C*      JOB I.                                                           *
C************************************************************************
        DO 215 P = 1, TYPE
        N = J2PROD(P)
        IF (N .EQ. 0) GO TO 215
205     IF (RJI .LT. RJOB(N) ) GO TO 206
           IF (N .EQ. JPROD1(P) ) GO TO 207
        N = JOB3(N)
        GO TO 205
206     IF (N .EQ. J2PROD(P) ) GO TO 215
        N = JOB4(N)
207     RJN = PJOB(N)
        IF (RJN .LT. RJOB(I) ) GO TO 210
C************************************************************************
C*      JOBS N AND I HAVE EQUAL DEFERRAL COSTS. IF JOB N CAN ALSO BE A   *
C*      PSEUDO PREDECESSOR OF I, BREAK THE TIE ARBITRARILY IN FAVOR OF   *
C*      THE LOWER PRODUCT NUMBER JOB - N.                                *
C************************************************************************
        K = J2PROD(TYPE + 1)
208     IF (K .EQ. I) GO TO 206
        IF (RJOB(K) .GT. RJN) GO TO 210
        K = JOB3(K)
        GO TO 208
210     NEXT = NPRED(N,1)
213     NEXT = NEXT + 1
        NPRED(N,1) = NEXT
        NPRED(N,NEXT) = I
215     CONTINUE
        IF (TYPE .EQ. NPROD - 1) GO TO 232
C************************************************************************
```

```
C*     EXAMINE HIGHER NUMBER PRODUCT STRINGS FOR PSEUDO SUCCESSOR JOBS   *
C*     TO JOB I.                                                          *
C*************************************************************************
216    TYPE = TYPE + 2
       DO 230 P = TYPE, NPROD
       N = J2PROD(P)
       IF (N .EQ. 0) GO TO 230
217    IF (FJI .LT. RJOB(N) ) GO TO 221
       IF (N .EQ. JPROD1(P) ) GO TO 225
       N = JOB3(N)
       GO TO 217
221    IF (N .EQ. J2PROD(P) ) GO TO 230
       N = JOB4(N)
225    NEXT = LPRED(N,1)
       IF (AVAL) .GT. JOB2(N) ) GO TO 228
       IF (NEXT .EQ. 1) GO TO 228
       DO 227 M = 2, NEXT
       IF ( JOB1( LPRED(N,M) ) .NE. TYPE - 1) GO TO 227
       LPRED(N,M) = I
       GO TO 230
227    CONTINUE
228    NEXT = NEXT + 1
       LPRED(N,1) = NEXT
       LPRED(N,NEXT) = I
230    CONTINUE
232    CONTINUE
C*************************************************************************
C*     PRINT OUT THE SET OF JOBS AND THEIR PSEUDO PREDECESSORS.          *
C*************************************************************************
       IF(.NOT.PRINT) GO TO 243
       WRITE (6,60)
       WRITE (6,234) TIME1, TIME2
234    FORMAT (//, 5X, 'PERIOD ', I5, '  THROUGH PERIOD ', I5, //)
       WRITE (6,235)
235    FORMAT (//, 10X, 'JOB', 10X, 'PSEUDO PREDESSORS', //)
       DO 242 I = 1, NJOB
       IF ( JOB2(I) .LT. ( ) GO TO 242
       IF ( JOB2(I) .GT. TIME2) GO TO 242
       LAST1 = LPRED(I,1)
       LAST2 = HPRED(I,1)
       IF (LAST1 .GT. 1) GO TO 236
       LAST1 = 2
       LPRED(I,2) = 0
236    IF (LAST2 .GT. 1) GO TO 237
       IF ( LPRED(I,2) .EQ. 0) GO TO 242
       WRITE (6,241) I, (LPRED(I,K), K = 2, LAST1)
       GO TO 242
237    IF ( LPRED(I,2) .EQ. 0) GO TO 240
       WRITE (6,241) I, (LPRED(I,K), K = 2, LAST1),
      *         (HPRED(I,K), K = 2, LAST2)
       GO TO 242
240    WRITE (6,241) I, ( HPRED(I,K), K = 2, LAST2)
241    FORMAT (//, 10X, I3, 5X, 20I5)
242    CONTINUE
243    L = LEV1
       T = TIME1
       TMAX=0
       JMT = JMACH(T)
       JMTM1 = JMT - MSCHED
C*************************************************************************
C*     THIS PORTION OF THE PROGRAM SEARCHES DOWN THROUGH THE TREE SO      *
C*     PLEV(J) = 0 FOR ALL J > 1.                                         *
C*                                                                        *
C*     FIND THE NEXT PRODUCT TYPE TO BE CONSIDERED AT LEVEL L. IF THERE   *
C*     IS A JOB OF THIS PRODUCT TO BE SCHEDULED, PLACE IT IN THE DUMMY    *
C*     SCHEDULE, UPDATE THE COST, AND CHECK IF THE COST IS WITHIN         *
C*     THE BOUND.                                                         *
C*************************************************************************
       I = JPROD1(PLEV(L) )
245    PLEV(L) = PLEV(L)+1
       IF(I.EQ.0) GO TO 316
       IF (T .LT. JOB2(I) ) GO TO 316
```

```
C***********************************************************************
C*    EXAMINE THE LIST OF PSEUDO PREDECESSORS OF JOB I FROM LOWER NUM- *
C*    BERED PRODUCTS. IF A PREDECESSOR IS AVAILABLE AND HAS NOT BEEN   *
C*    SCHEDULED YET, THE CURRENT SCHEDULE CANNOT BE OPTIMAL.           *
C***********************************************************************
      LAST = LPRED(I,1)
      IF (LAST .EQ. 1) GO TO 255
      DO 250 JP = 2, LAST
      TESTJ = LPRED(I,JP)
         IF (SJOB(TESTJ) .GT. 0) GO TO 250
      IF (JOB2(TESTJ) .GT. T) GO TO 250
      GO TO 316
250   CONTINUE
C***********************************************************************
C*    ATTEMPT TO ASSIGN JOB I TO LEVEL L                               *
C***********************************************************************
255   DSCHED(L) = I
      COSTI = COSTI + RJOB(I) * T
      BTI = BT(I)
      NBS(BTI) = NBS(BTI) + 1
      LNOW = L
      IF (L .EQ. LEV2) GO TO 300
2222  IF(COSTI+LBND(L+1).LT.COST) GO TO 258
C***********************************************************************
C*    ASSIGNING JOB I TO THIS LEVEL WOULD DRIVE THE COST ABOVE THE     *
C*    CURRENT "BEST COST". TRY AN ALTERNATIVE PRODUCT.                 *
C***********************************************************************
257   IF(T .EQ. TIME2) GO TO 430
      COSTI = COSTI - RJOB(I) * T
      DSCHED(L) = 0
      NBS(BTI) = NBS(BTI) - 1
      GO TO 316
258   SJOB(I) = T
      JPRODT(PLEV(L)) = JOB4(I)
      LMAX = 0
      IF (L .GT. LEV1) LMAX = MAXL(L - 1)
      IF (LMAX .LT. BJOB(I) ) LMAX = BJOB(I)
      MAXL(L) = LMAX
      IF(TMAX .GE. BTI) GO TO 259
      TMAX = BTI
      MAXT(T) = TMAX
      GO TO 269
C***********************************************************************
C*    IF THE NEXT LEVEL IS IN A LATER TIME PERIOD EXAMINE THE LIST OF  *
C*    PSEUDO PREDECESSORS FROM HIGHER NUMBERED PRODUCTS FOR THE JOBS IN*
C*    THIS TIME PERIOD.                                                *
C***********************************************************************
259   IF(T .LT. TMAX) GO TO 260
      IF(T .EQ. BTI) GO TO 260
      IF(L .EQ. JMTM1 + NBS(T)) GO TO 375
260   IF(L .LT. JNT) GO TO 315
      LIM1 = JMTM1 + 1
      LIM2 = LIM1 + MACH(T) - 1
      COUNT = 0
      DO 270 N = LIM1, LIM2
      I = DSCHED(N)
      LAST = HPRED(I,1)
      IF (LAST .EQ. 1) GO TO 270
      DO 265 JP = 2, LAST
      TESTJ = HPRED(I,JP)
      IF (SJOB(TESTJ) .GT. 0) GO TO 265
      IF (JOB2(TESTJ) .GT. T) GO TO 265
      COUNT = COUNT + 1
265   CONTINUE
      IF (COUNT .EQ. 0) GO TO 270
      COUNT1 = COUNT - 1
      IF(COUNT .GT. LIM2-N+1) GO TO 273
      GO TO 275
270   CONTINUE
      GO TO 285
273   COUNT1 = LIM2 - N
275   IF (COUNT1 .EQ. 0) GO TO 261
```

```
      DO 280 MI = 1 , COUNT1
      M = LIM2 - MI + 1
      I = DSCHED(M)
      DSCHED(M) = 0
      PLEV(M) = 0
      JPRED1( JOB1(I) ) = 1
      COST1 = COST1 - PJOB(I) * T
      SJOB(I) = 0
      MAXL(M) = 0
      BTI = BT(I)
      NOS(BTI) = NBS(BTI) - 1
280   CONTINUE
      TMAX = (MAXL(M-1) + MSCHED-1) / MSCHED
      MAXT(T) = TMAX
281   L = LIM2 - COUNT1 + 1
      GO TO 317
C**********************************************************************
C*    MOVE TO THE NEXT LOWER LEVEL ON THE TREE. DETERMINE IF THE PROG- *
C*    LEM IS TAKING AN EXCESSIVE AMOUNT OF TIME.                       *
C**********************************************************************
285   T = T + 1
      JMT = JMACH(T)
      JMTM1 = JMT - MSCHED
      MAXT(T) = MAXT(T-1)
C**********************************************************************
C*    DETERMINE IF A JOB SHOULD BE ASSIGNED TO THIS LEVEL. IF NOT,     *
C*    EVALUATE THE COST PRIOR TO MOVING TO A LOWER LEVEL. OTHERWISE,   *
C*    ASSIGN A JOB TO THIS LEVEL.                                      *
C**********************************************************************
      BJ = TTIME(0)
      IF (BJ .GT. RUSS) GO TO 290
      GO TO 315
290   WRITE (6,291)
291   FORMAT (///, 5X, ' THE CURRENT INCOMPLETE SCHEDULE IS')
      WRITE (6,294)
294   FORMAT (///, ' TIME PERIOD', 25X, 'MACHINE')
      WRITE (6,296) (I, I = 1, MSCHED)
296   FORMAT (/, 25X, 14I5)
      WRITE (6,298)
298   FORMAT (//)
      LOW = LEV1
      HIGH = LOW + MSCHED - 1
      DO 310 I = TIME1, TIME2
      WRITE (6,300) I, (DSCHED(L), L = LOW, HIGH)
300   FORMAT (5X, I5, 15X, 14I5)
      IF (HIGH .GE. LEV2) GO TO 312
      LOW = LOW + MSCHED
      HIGH = HIGH + MSCHED
310   CONTINUE
312   WRITE (6,314) COST1, COST
314   FORMAT (//, ' THE SCHEDULE COST FOR THIS INCOMPLETE SCHEDULE IS',
     *        I9, /, ' THE BEST SCHEDULE COST TO DATA FOR A COMPLETE ',
     *        'SCHEDULE IS', I9)
      IF(.NOT. BJZ) RETURN
      RUSS = RUSS + RP
C**********************************************************************
C*    DETERMINE IF A JOB SHOULD BE ASSIGNED TO THIS LEVEL. IF NOT,     *
C*    EVALUATE THE COST PRIOR TO MOVING TO A LOWER LEVEL. OTHERWISE,   *
C*    ASSIGN A JOB TO THIS LEVEL.                                      *
C**********************************************************************
315   L = L + 1
      MAXL(L) = MAXL(L - 1)
      IF (DISKD(L,I) .EQ. 0) GO TO 260
      MAXL(L)=0
      IF(L .GT. JMTM1+1) PLEV(L) = PLEV(L-1)+1
      GO TO 245
C**********************************************************************
C*    ANY RESULTING SCHEDULE WHICH AGREES WITH LEVELS 1 THROUGH L-1    *
C*    AND HAS LEVEL L EMPTY WOULD BE WORSE THAN AN EXISTING BOUND.     *
C**********************************************************************
316   IF(PLEV(L).LT.NPPOD) GO TO 245
317   IF (L .EQ. LEV1) GO TO 434
```

```
      PLEV(L) = 0
C***********************************************************************
C*    THIS PORTION WORKS ITS WAY BACK UP THROUGH TREE AND TRIES TO     *
C*    CHANGE AN EXISTING OR STOPPED SCHEDULE. WE ASSUME THAT A JOB IS  *
C*    AVAILABLE IN THE FIRST PERIOD                                    *
C***********************************************************************
318   L = L-1
      MAXL(L) = 0
      IF (DISKD(L,1) .EQ. 0) GO TO 318
      IF(L .GT. JMIMI) GO TO 319
      MAXT(T) = 0
      T = T - 1
      JMT = JMACH(T)
      JMIMI = JMT - MSCHED
      TMAX = MAXT(T)
319   I = DSCHED(L)
      SJOB(I) = 0
      JPROD1( JOB1(I) ) = I
      COST1 = COST1 - RJOB(I) * T
      DSCHED(L) = 0
      BTI = BT(I)
      NOS(BTI) = NOS(BTI) - 1
      IF(TMAX .GT. BTI) GO TO 322
      IF(NOS(BTI) .GT. 0) GO TO 322
      IF(L .GT. JMIMI + 1) GO TO 321
      IF(L .GT. LEV1) GO TO 320
      TMAX = 0
      MAXT(T) = 0
      GO TO 322
320   TMAX = MAXT(T - 1)
      MAXT(T)=TMAX
      GO TO 322
321   TMAX = (MAXL(L - 1) + MSCHED - 1) / MSCHED
      MAXT(T) = TMAX
322   IF(PLEV(L).LT.NPROD) GO TO 324
C***********************************************************************
C*    IF WE HAVE REACHED THE TOP LEVEL OF THE TREE, THE SEARCH IS      *
C*    FINISHED. OTHERWISE, MOVE TO THE NEXT HIGHER LEVEL AND CONTINUE  *
C*    THE SEARCH.                                                      *
C***********************************************************************
      GO TO 317
324   PLEV(L) = PLEV(L)+1
      I = JPROD1(PLEV(L) )
      IF(I.EQ.0) GO TO 322
      IF (T .LT. JOB2(I) ) GO TO 322
C***********************************************************************
C*    EXAMINE THE LIST OF PSEUDO PREDECESSORS OF JOB I FROM LOWER NUM- *
C*    BERED PRODUCTS. IF A PREDECESSOR IS AVAILABLE AND HAS NOT BEEN   *
C*    SCHEDULED YET, THE CURRENT SCHEDULE CANNOT BE OPTIMAL.           *
C***********************************************************************
      LAST = LPRED(I,1)
      IF (LAST .EQ. 1) GO TO 255
      DO 350 JP = 2, LAST
      TESTJ = IPRED(I,JP)
      IF (SJOB(TESTJ) .GT. 0) GO TO 350
      IF (JOB2(TESTJ) .GT. T) GO TO 350
      GO TO 322
350   CONTINUE
      GO TO 255
C***********************************************************************
C*    THE REMAINING LEVELS MUST AGREE WITH THOSE IN THE CURRENT BEST   *
C*    SOLUTION. EVALUATE THE COST OF THE ENTIRE SCHEDULE.              *
C***********************************************************************
375   CT = 0
      LIM1 = JMIMI + 1
      LIM2 = JMIMI + MACH(T)
      IF(T .EQ. TIME2) LIM2 = LEV2
      DO 376 LK = LIM1,LIM2
      IK = BLEV(LK)
      IF(SJOB(IK) .GT. 0) CT = CT + RJOB(IK) * T
376   CONTINUE
377   CT = CT + C(T-1)
```

```
      IF(COSTI.LE.CT) GO TO 378
      JPRODI(BLEV(L)) = 1
      SJOB(I) = 0
      MAXL(L) = 0
      GO TO 257
378   L1 = L + 1
      IF(L .EQ. LIM2) GO TO 383
      LL = LIM1
      DO 380 I = L1,LIM2
      DO 379 LK = LL,LIM2
      IK = BLEV(LK)
      IF(SJOB(IK) .GT. 0) GO TO 379
      GO TO 382
379   CONTINUE
382   DSCHED(L) = IK
      COSTI = COSTI + RJOB(IK) * T
      LL = LK + 1
380   CONTINUE
      SLIM = LIM2 - 1
      NLIM = LIM2
      DO 1001 MSORT = LIM1,SLIM
      NLIM = NLIM - 1
      DO 1000 LSORT = LIM1,NLIM
      IF (DSCHED(LSORT).LT.DSCHED(LSORT+1)) GO TO 1000
      JSORT = DSCHED (LSORT)
      DSCHED(LSORT) = DSCHED(LSORT+1)
      DSCHED(LSORT+1) = JSORT
1000  CONTINUE
1001  CONTINUE
383   COSTI = C(TIME2) - C(T) + COSTI
      IF(T .LT. TIME2) GO TO 384
      LEV=JMACH(TIME2)-MSCHED+1
      GO TO 390
384   LL = LIM2 + 1
      DO 381 L = LL,LEV2
      DSCHED(L) = BLEV(L)
381   CONTINUE
C***********************************************************************
C*    RECORD THE LEVEL EACH JOB IS ASSIGNED TO, AND THE COST OF THIS   *
C*    SCHEDULE FROM THE BEGINNING OF THE SEPARABLE SECTION TO EACH     *
C*    LEVEL.                                                           *
C***********************************************************************
390   IF (COSTI .GT. COST) GO TO 430
      T = TIME1
      C(T) = 0
      JMT = JMACH(T)
      NBS(T) = 0
      MAXT(T) = T
      DO 410 L = LEV1, LEV2
      JOB = DSCHED(L)
      IF (JOB .EQ. 0) GO TO 405
      MAXL(L) = L
      BLEV(L) = JOB
      BJOB(JOB) = L
      C(T) = C(T) + RJOB(JOB) * T
      BT(JOB) = T
      SJOB(JOB) = T
      NBS(T) = NBS(T) + 1
      GO TO 406
405   MAXL(L) = MAXL(L-1)
406   IF(L .LT. JMT) GO TO 410
      T = T + 1
      C(T) = C(T-1)
      JMT = JMACH(T)
      NBS(T) = 0
      MAXT(T) = T
410   CONTINUE
      LNOW = LEV2
      JMTM1 = JMT - MSCHED
C***********************************************************************
C*    THIS IS THE BEST COMPLETED SCHEDULE FOUND SO FAR. LOAD IT INTO   *
C*    SCHED( , ,3).                                                    *
```

```
C***********************************************************************
      COST = COST1
      DO 415 J = LEV1, LEV2
      DISKD(J,3) = OSCHED(J)
415   CONTINUE
C***********************************************************************
C*    BACK UP TO THE LEVEL OF THE LAST MACHINE OF THE NEXT TO LAST     *
C*    TIME PERIOD.                                                     *
C***********************************************************************
430   DO 432 J = L1, LNOW
      K = LNOW - J + L1
      I = OSCHED(K)
      MAXL(K) = 0
      IF (I .EQ. 0) GO TO 431
      BTI = BT(I)
      NBS(BTI) = NBS(BTI) - 1
      COST1 = COST1 - RJOB(I) * T
      JPROD1( JOB1(I) ) = 1
      OSCHED(K) = 0
      SJOB(I) = 0
      MLEV(K) = 0
431   IF(K .GT. JMTM1 + 1) GO TO 432
      MAXT(T) = 0
      T = T - 1
      JMT = JMACH(T)
      JMTM1 = JMT - MSCHED
      TMAX = MAXT(T)
432   CONTINUE
      L = L1
      L1 = JMACH(TIME2) - MSCHED + 1
      GO TO 316
C***********************************************************************
C*    THIS IS THE OPTIMAL SCHEDULE FOR THIS TIME SECTION. IF IT IS THE *
C*    LAST TIME SECTION THE ENTIRE SCHEDULE IS OPTIMAL. OTHERWISE,     *
C*    MOVE ON TO THE NEXT TIME SECTION.                                *
C***********************************************************************
434   COPT = COPT + COST
      IF (COST .LT. COST2) GO TO 439
      DO 437 I = LEV1, LEV2
      DISKD(I,3) = DISKD(I,NNUM)
437   CONTINUE
439   DO 441 J = TIME1,TIME2
      MACHJ = MACH(J)
      DO 440 I = 1, MACHJ
      JOB = SCHED(I,J,3)
      IF(JOB .EQ. 0) GO TO 441
      SJOB(JOB) = J
440   CONTINUE
441   CONTINUE
      IF (TIME2 .EQ. TIME) GO TO 450
C***********************************************************************
C*    REMOVE THOSE JOBS WHICH HAVE BEEN SCHEDULED FOR THIS SEPARABLE   *
C*    PROBLEM SECTION FROM THE JPROD VECTOR.                           *
C***********************************************************************
      IF (TIME1 .EQ. TIME2) GO TO 444
      DO 443 J = TIME1, T2
      MACHJ = MACH(J)
      DO 443 I = 1, MACHJ
      NJ = SCHED(I,J,3)
      IF (JPROD1(JOB1(NJ)).LE.NJ) JPROD1(JOB1(NJ))=JOB4(NJ)
443   CONTINUE
444   DO 445 L = L1, LEV2
      NJ = DISKD(L,3)
      IF ( JPROD1( JOB1(NJ) ) .LE. NJ) JPROD1( JOB1(NJ) ) = JOB4(NJ)
445   CONTINUE
C***********************************************************************
C*    FIND THE BEGINNING OF THE NEXT SEPARABLE PROBLEM SECTION         *
C***********************************************************************
      T = TIME2 + 1
      DO 447 TIME1 = T, TIME
      IF (SCHED(1,TIME1,1) .GT. 0) GO TO 65
447   CONTINUE

C***********************************************************************
C*    THE OPTIMAL SOLUTION HAS BEEN FOUND FOR THE OVERALL PROBLEM.     *
C*    SUBTRACT OFF THE PORTION OF THE DEFERRAL COST ASSOCIATED WITH THE*
C*    TIME BEFORE A JOB IS AVAILABLE.                                  *
C***********************************************************************
450   DO 475 I = 1, NJOB
475   COPT = COPT - DJOB(I)
      RETURN
      END
```

```
      SUBROUTINE LOWBND
C***********************************************************************
C***********************************************************************
C*                                                                     *
C*    SUBROUTINE LOWBND                                                *
C*    THIS SUBROUTINE USES A GREEDY SCHEDULING ALGORITHM WITH AVAIL-   *
C*    ABILITY TIMES. THE COSTS ARE USED AS LOWER BOUNDS FOR THE 'BEST  *
C*    FEASIBLE SOLUTION THAT MAY BE OBTAINED FROM THIS POINT'. FOR     *
C*    THIS ALGORITHM PRECEDENCE CONSTRAINTS ARE IGNORED, EXCEPT THAT   *
C*    A JOB IS NOT SCHEDULED IN THE FIRST MACHINE OF ITS PREDECESSOR'S *
C*    AVAILABILITY TIME IF IT HAS THE SAME AVAILABILITY TIME AS ITS    *
C*    PREDECESSOR.                                                     *
C*                                                                     *
C***********************************************************************
C***********************************************************************
      IMPLICIT INTEGER*4 (A-Z)
      COMMON LBND(1000),TLBND(1000),J1PROD(20),JPROD1(20),JPROD2(20),
     *       MACH(200),JMACH(200),NMACH(200),RJOB(200),SJOB(200),
     *       JOB1(200),JOB2(200),JOB3(200),JOB4(200),DJOB(200),
     *       SCHED(5,200,3),TIME,NJOB,NPROD,HCOST,COPT,MSCHED,TIME1,
     *       TIME2,PRINT
      LOGICAL PRINT
      DIMENSION DISKD(1000,3)
      EQUIVALENCE (DISKD(1,1), SCHED(1,1,1))
C***********************************************************************
C*    CONSTRUCT THE JOB SCHEDULE VIA THE GREEDY SCHEDULING ALGORITHM   *
C***********************************************************************
      NLEV = 0
      DO 210 I = 1, TIME
      NLEV = NLEV + MSCHED
      JMACH(I) = NLEV
210   CONTINUE
      DO 220 I = 1, NJOB
      SJOB(I) = 0
220   CONTINUE
      T = 1
      DO 240 L = 1, NLEV
      IF (L .GT. JMACH(T) ) T = T + 1
      LBND(L) = 0
      IF (DISKD(L,1) .EQ. 0) GO TO 240
      COST = 0
      DO 230 I = 1, NJOB
      IF (SJOB(I) .EQ. 1) GO TO 230
      IF (T .LT. JOB2(I) ) GO TO 230
      J = JOB3(I)
      IF (J .EQ. 0) GO TO 225
      IF (T .GT. JOB2(J) ) GO TO 225
      IF (L .EQ. JMACH(T) - MSCHED + 1) GO TO 230
225   IF (COST .GE. RJOB(I) ) GO TO 230
      COST = RJOB(I)
      JNUM = I
230   CONTINUE
      LBND(L) = JNUM
      SJOB(JNUM) = 1
240   CONTINUE
C***********************************************************************
C*    PRINT OUT THE JOB SCHEDULE CONSTRUCTED VIA THE GREEDY ALGORITHM  *
C*    AND SUBTRACT OFF THE PORTION OF THE DEFERRAL COST WHICH IS ASSO- *
C*    CIATED WITH THE TIME BEFORE THE JOB IS AVAILABLE.                *
C***********************************************************************
      IF (.NOT. PRINT) GO TO 446
      WRITE (6,442)
442   FORMAT (/////, ' THE JOB SCHEDULE CONSTRUCTED BY ', /,
     *        ' THE GREEDY (LOWER BOUND) ALGORITHM IS')
443   WRITE (6,444)
444   FORMAT (///, ' TIME PERIOD',20X,'MACHINE')
      WRITE (6,445) (K, K = 1, MSCHED)
```

```
445   FORMAT (/, 25X, 21I5)
      WRITE (6,446)
446   FORMAT (//)
448   LOW = 1 - MSCHED
      DO 460 J = 1, TIME
      LOW = LOW + MSCHED
      HIGH = LOW + MSCHED - 1
      DO 450 K = LOW, HIGH
      TLBND(K) = 0
      I = LBND(K)
      IF (I .EQ. 0) GO TO 450
      DJOB(I) = RJOB(I) * JOB2(I)
      TLBND(K) = DJOB(I)
450   CONTINUE
      HIGH = LOW + MACH(J) - 1
      IF (PRINT) WRITE (6,461) J, (LBND(K), K = LOW, HIGH)
460   CONTINUE
461   FORMAT (5X, I5, (5X, 21I5))
C*****************************************************************************
C*    COMPUTE THE COST OF PRODUCTION UNDER THE GREEDY SCHEDULE FROM          *
C*    EACH POINT IN THE SCHEDULE TO THE TIME HORIZON.                        *
C*                                                                           *
C*    DETERMINE THE END POINT OF THE NEXT TIME SECTION                       *
C*****************************************************************************
462   LEV1 = NLEV + 1
463   LEV0 = LEV1 - 1
      IF (LEV0 .EQ. JMACH(T) - MSCHED) T = T - 1
      LEV2 = LEV0
      IF (DISKD(LEV0,1) .GT. 0) GO TO 465
      DO 464 I = 2, LEV0
      LEV2 = LEV0 - I + 1
      IF (LEV2 .EQ. JMACH(T) - MSCHED) T = T - 1
464   IF (DISKD(LEV2,1) .GT. 0) GO TO 465
      GO TO 484
465   IF (LEV2 .EQ. 1) GO TO 469
      IF (T .EQ. 1) GO TO 467
C*****************************************************************************
C*    FIND THE BEGINING OF THIS TIME SECTION                                 *
C*****************************************************************************
      DO 466 I = 2, T
      TNEW = T - I + 1
      IF (DISKD(JMACH(T) - I * MSCHED + MACH(TNEW), 1) .EQ. 0) GO TO 468
466   CONTINUE
467   LEV1 = 1
      GO TO 473
468   LEV1 = JMACH(TNEW) + 1
      IF (LEV2 .GT. LEV1) GO TO 473
C*****************************************************************************
C*    CALCULATE THE SCHEDULE COST FROM EACH POINT IN THIS TIME SECTION *
C*    TO THE TIME HORIZON. NOTE THAT ANY JOB SCHEDULED IN LEV1 MUST BE *
C*    AT ITS AVAILABILITY TIME AND THUS HAS A ZERO COST.                     *
C*****************************************************************************
      LBND(LEV2) = RJOB(LBND(LEV2) ) * T
      TLBND(LEV2) = LBND(LEV2) - TLBND(LEV2)
      GO TO 463
469   LBND(1) = RJOB(LBND(1) )
      TLBND(1) = LBND(1) - TLBND(1)
      GO TO 484
473   J = LBND(LEV2)
      LBND(LEV2) = RJOB(J) * T
      TLBND(LEV2) = LBND(LEV2) - TLBND(LEV2)
      K = LEV2 - LEV1
      DO 480 I = 1, K
      L = LEV2 - I
      IF (L .EQ. JMACH(T) - MSCHED) T = T - 1
      J = LBND(L)
      IF (J .EQ. 0) GO TO 475
      LBND(L) = LBND(L + 1) + RJOB(J) * T
      TLBND(L) = TLBND(L + 1) + RJOB(J) * T -TLBND(L)
      GO TO 480
475   LBND(L) = LBND(L + 1)
      TLBND(L) = TLBND(L + 1)
```

```
480    CONTINUE
       IF (LEVEL .GT. 1) GO TO 483
C*********************************************************************
C*     PRINT OUT THE GREEDY JOB SCHEDULE'S SCHEDULE COSTS             *
C*********************************************************************
486    IF (.NOT. PRINT) RETURN
       WRITE (6,495)
495    FORMAT (///, ' THE COST OF PRODUCTION FROM EACH POINT IN THE',
     *         ' SCHEDULE', /, ' TO THE END OF THE TIME HORIZON UNDER',
     *         ' THE GREEDY SCHEDULE IS')
       WRITE (6,497)
497    FORMAT (///, ' TIME PERIOD', 20X, 'MACHINE')
       WRITE (6,498) (K, K = 1, MSCHED)
498    FORMAT (/, 25X, 13I8)
       WRITE (6,496)
       LOW = 1 - MSCHED
       DO 505 J = 1, TIME
       LOW = LOW + MSCHED
       HIGH = LOW + MSCHED - 1
505    WRITE (6,506) J, (CLBND(K), K = LOW, HIGH)
506    FORMAT (5X, I5, 13X, 13F8)
       RETURN
       END


       SUBROUTINE GENER(SEED,ALIMIT)
C*********************************************************************
C*********************************************************************
C*                                                                   *
C*     SUBROUTINE GENER                                              *
C*     THIS SUBROUTINE GENERATES A SET OF JOBS TO BE SCHEDULED GIVEN THE*
C*     TOTAL NUMBER OF JOBS, THE NUMBER OF PRODUCTS, AND THE TIME HOR-  *
C*     IZON DESIRED.                                                 *
C*                                                                   *
C*********************************************************************
C*********************************************************************
       IMPLICIT INTEGER*4 (A-Z)
       INTEGER*4 SEED
       REAL URAND,ALIMIT
       COMMON LBND(1000),TLBND(1000),JPROD(20),JPROD1(20),JPROD2(20),
     *        MACH(200),JMACH(200),NMACH(200),RJOB(200),SJOB(200),
     *        JOB1(200),JOB2(200),JOB3(200),JOB4(200),SJOB(200),
     *        SCHED(5,200,5),TIME,NJOB,NPROD,HCOST,COPT,MSCHED,TIME1,
     *        TIME2,PRINT
       LOGICAL PRINT
       DIMENSION JOB(25)
C*********************************************************************
C*     PRINT OUT THE PROBLEM CHARACTERISTICS                         *
C*********************************************************************
       ALAST = TIME * ALIMIT
       WRITE (6,10) ALAST
10     FORMAT (/, ' JOBS HAVE AVAILABILITY TIMES FROM PERIOD  1 ',
     *         ' TO PERIOD', I5)
       WRITE (6,13) SEED
13     FORMAT ( /, ' THE RANDOM NUMBER GENERATOR SEED IS', I20)
       WRITE (6,14)
14     FORMAT (/////, ' JOB NUMBER', 14X, 'JOB MATRIX', 13X,
     *         'RJOB VECTOR', ///)
C*********************************************************************
C*     DETERMINE THE NUMBER OF JOBS IN EACH PRODUCT STRING           *
C*********************************************************************
       PSTART = 1
       DO 40 I = 1, NPROD
       JOB(I) = 1
40     CONTINUE
       NUM = NJOB - NPROD
       DO 50 I = 1, NUM
       P = 1. + URAND(SEED) * NPROD
       JOB(P) = JOB(P) + 1
50     CONTINUE
C*********************************************************************
C*     THE ALGORITHM REQUIRES THAT AT LEAST ONE JOB BE AVAILABLE DURING *
C*     PERIOD 1, SO FORCE JOB 1 TO BE AVAILABLE DURING THE FIRST PERIOD *
C*********************************************************************
       JOB1(1) = 1
       JOB2(1) = 1
       JOB3(1) = 0
       IF ( JOB(1) .EQ. 1) GO TO 60
       JOB4(1) = 2
       JOB3(2) = 1
       GO TO 65
60     JOB4(1) = 0
       PSTART = 2
65     RJOB(1) = 1. + URAND(SEED) * 50
       JOB(1) = JOB(1) - 1
C*********************************************************************
C*     DETERMINE THE CHARACTERISTICS OF EACH JOB                     *
C*********************************************************************
       J = 2
       DO 100 P = PSTART, NPROD
       IF (P .EQ. 1) GO TO 68
       JOB3(J) = 0
```

```
68    LIMIT = JOB(P)
C***********************************************************************
C*    DETERMINE THE RANGE FOR EACH JOB'S AVAILABILITY TIME             *
C***********************************************************************
      DELT = ALAST / JOB(P)
      IF (DELT .LT. 1) DELT = 1
      NLAST = 0
      LAST1 = 1
      DO 95 I = 1, LIMIT
      JOB1(J) = P
      NLAST = NLAST + DELT
      LAST1 = NLAST - (NLAST - LAST1) * URAND(SEED)
      JOB2(J) = LAST1
      IF (I .EQ. 1)  GO TO 70
      JOB3(J) = J - 1
      JOB4(J - 1) = J
70    JOB4(J) = 0
      RJOB(J) = 1. + URAND(SEED) * 50
      J = J + 1
95    CONTINUE
100   CONTINUE
C***********************************************************************
C*    PRINT OUT THE GENERATED JOBS AND THEIR CHARACTERISTICS.          *
C***********************************************************************
      DO 120 J = 1, NJOB
      WRITE (6,110) J, JOB1(J), JOB2(J), JOB3(J), JOB4(J), RJOB(J)
110   FORMAT (I5, 4I10, 2X, I5)
120   CONTINUE
      RETURN
      END

      SUBROUTINE SCHED1(FLAG,FLAG1,FLAG2)
C***********************************************************************
C***********************************************************************
C*    SUBROUTINE SCHED1                                               **
C*    IN SCHEDULING EACH CANDIDATE JOB IS WEIGHTED BY THE DEFERRAL    **
C*    COSTS OF THOSE JOBS OF ITS PRODUCT WHICH WOULD BE PUSHED LATER  **
C*    ONE PERIOD IF THAT JOB IS NOT SCHEDULED ON THE FIRST AVAILABLE  **
C*    MACHINE.  THE DEFERRAL COST OF THE CANDIDATE JOB IS INCLUDED    **
C*    ONLY IF IT TOO WOULD BE PUSHED LATER ONE PERIOD                 **
C***********************************************************************
C***********************************************************************
      IMPLICIT INTEGER*4 (A-Z)
      COMMON LBND(1000),TLLND(1000),J1PROD(20),JPROD1(20),JPROD2(20),
     *        MACH(200),JMACH(200),NMACH(200),RJOB(200),SJOB(200),
     *        JOB1(200),JOB2(200),JOB3(200),JOB4(200),DJOB(200),
     *        SCHED(5,200,3),TIME,NJOB,NPROD,HCOST,COPT,MSCHED,TIME1,
     *        TIME2,PRINT
      LOGICAL PRINT,FLAG,FLAG1,FLAG2
      COPT = 0
      ITIME = 1
      DO 83 I = 1,TIME
      NMACH(I) = MACH(I)
83    CONTINUE
      DO 85 IPROD = 1,NPROD
      JPROD1(IPROD) = J1PROD(IPROD)
85    CONTINUE
90    JNUM = 0
      COST = -1
      COST1 = 0
      IDONE = 0
C***********************************************************************
C*    SEARCH FOR A JOB TO BE SCHEDULED                                 *
C***********************************************************************
C
      DO 110 IPROD = 1,NPROD
      I = JPROD1(IPROD)
      IDUM = I
      IF(I.GT.0) GO TO 103
      IDONE = IDONE+1
      GO TO 110
103   IF (ITIME .LT. JOB2(I)) GO TO 110
      DO 104 JTIME = ITIME,TIME
104   JMACH(JTIME) = NMACH(JTIME)
C***********************************************************************
C*    ASSUME THAT JOB I IS NOT SCHEDULED IN MACHINE JMACH(ITIME), THEN *
C*    EXAMINE THE SEQUENCE OF 'FORCED' SUCCESSORS OF JOB I             *
C***********************************************************************
      JTIME= ITIME
      JMACH(JTIME) = JMACH(JTIME)-1
      IF(JMACH(JTIME).GT.0) GO TO 106
      JTIME = JTIME+1
      GO TO 107
105   IDUM = JOB4(IDUM)
      IF(IDUM.EQ.0) GO TO 109
      IF (JTIME .LT. JOB2(IDUM) ) GO TO 109
106   JMACH(JTIME) = JMACH(JTIME)-1
      IF(JMACH(JTIME).GT.0) GO TO 105
      JTIME = JTIME+1
      IDUM = JOB4(IDUM)
      IF(IDUM.EQ.0) GO TO 109
      IF (JTIME .LE. JOB2(IDUM) ) GO TO 109
C***********************************************************************
C*    IF DELAYING JOB I ONE MACHINE DRIVES IT'S SUCCESSORS INFEASIBLE, *
C*    SCHEDULE JOB I IN MACHINE JMACH(ITIME). OTHERWISE, COMPARE THE   *
C*    COST OF DELAYING JOB I WITH THE COST OF DELAYING THE OTHER PROD- *
C*    UCTS.
C***********************************************************************
```

```
107   IF(JTIME.LE.TIME) GO TO 108
      JNUM = I
      GO TO 121
108   COST1 = COST1+RJOB(IDUM)
      GO TO 106
109   IF(COST.GE.COST1) GO TO 110
      COST = COST1
      JNUM = I
110   CONTINUE
C*********************************************************************
C*    JOB JNUM HAS BEEN CHOSEN TO BE SCHEDULED.  PLACE IT IN THE JOB  *
C*    SCHEDULE AND COMPUTE THE NEW SCHEDULE COST.                     *
C*********************************************************************
      IF(JNUM.GT.0) GO TO 121
      IF(IDONE.EQ.NPROD) GO TO 130
      ITIME = ITIME+1
      GO TO 128
121   SCHED( MACH(ITIME) - NMACH(ITIME) + 1, ITIME, 1) = JNUM
      JPROD2(JOB1(JNUM) ) = JOB4(JNUM)
      COPT = COPT + (ITIME - JOB2(JNUM) ) * RJOB(JNUM)
      NMACH(ITIME) = NMACH(ITIME)-1
      IF(NMACH(ITIME).EQ.0) ITIME = ITIME+1
128   IF(ITIME.GT.TIME) GO TO 130
      GO TO 99
C*********************************************************************
C*    THE SCHEDULE HAS BEEN COMPLETED. ATTEMPT TO IMPROVE IT VIA PAIR- *
C*    WISE JOB INTERCHANGES. THEN RETURN.                              *
C*********************************************************************
130   IF(FLAG) CALL PAIR(1,COPT,FLAG1,FLAG2)
      IF(COPT.LT.HCOST) HCOST = COPT
      RETURN
      END


C*********************************************************************
C*********************************************************************
C*                                                                    *
C*    FUNCTION URAND                                                  *
C*    THIS FUNCTION GENERATES RANDOM NUMBERS BETWEEN ZERO AND ONE. THE *
C*    SEQUENCE OF RANDOM NUMBERS IS DETERMINED BY THE SEED VALUE USED *
C*    INITIALLY - IX.                                                 *
C*                                                                    *
C*********************************************************************
C*********************************************************************
      FUNCTION URAND(IX)
      IX = IX * 65539
      IF (IX) 1, 2, 2
1     IX = IX + 2147483647 + 1
2     URAND = IX * .4656613E-9
      RETURN
      END
```

```
      SUBROUTINE SCHED2(FLAG,FLAG1,FLAG2)
C************************************************************************
C************************************************************************
C     SUBROUTINE SCHED2                                                 *
C*    JOBS ARE SCHEDULED BY SUBSTRINGS USING AN ADAPTATION OF SIDNEY'S *
C*    ALGORITHM OF RATIOS                                               *
C************************************************************************
C************************************************************************
      IMPLICIT INTEGER*4 (A-Z)
      COMMON LBND(1000),TLBND(1000),JIPROD(20),JPROD1(20),JPROD2(20),
     *       MACH(200),JMACH(200),NMACH(200),PJOB(200),SJOB(200),
     *       JOB1(200),JOB2(200),JOB3(200),JOB4(200),DJOB(200),
     *       SCHED(5,200,3),TIME,NJOB,NPROD,HCOST,COPT,MSCHED,TIME1,
     *       TIME2,PRINT
      LOGICAL PRINT,FLAG,FLAG1,FLAG2
      REAL COST,COST1,COST2
      COPT = 0
      ITIME = 1
      DO 152 I = 1,NPROD
      JPROD1(I) = JIPROD(I)
152   CONTINUE
      DO 153 I = 1,TIME
      NMACH(I) = MACH(I)
153   CONTINUE
C************************************************************************
C*    SEARCH FOR THE JOB TO BE SCHEDULED. DETERMINE THE SUBSTRING OF   *
C*    JOBS WHICH WOULD BE SCHEDULED IF THE FIRST JOB OF PRODUCT IPROD  *
C*    IS SCHEDULED IN MACHINE JMACH(ITIME).                             *
C************************************************************************
155   DO 165 IPROD = 1,NPROD
      I = JPROD1(IPROD)
      IF(I.EQ.0) GO TO 165
      IF (ITIME .LT. JOB2(I) ) GO TO 165
      JPROD2(IPROD) = I
      DO 160 JTIME = ITIME,TIME
      JMACH(JTIME) = NMACH(JTIME)
160    CONTINUE
      IDUM = I
      JTIME = ITIME
163   IF (JOB4(IDUM) .EQ. 0) GO TO 155
      JMACH(JTIME) = JMACH(JTIME)-1
      IF(JMACH(JTIME).EQ.0) JTIME = JTIME+1
      IDUM = JOB4(IDUM)
      IF (JTIME .LT. JOB2(IDUM)) GO TO 165
      JPROD2(IPROD) = IDUM
      GO TO 163
165   CONTINUE
C************************************************************************
C*    COMPUTE THE SCHEDULE COST OF ASSIGNING THE SUBSTRING OF JOBS FOR *
C*    EACH PRODUCT. DETERMINE THE SUBSTRING WITH THE LOWEST TOTAL COST *
C*    TO TOTAL PRODUCTION TIME RATIO.                                   *
C************************************************************************
      COST = 0.0
      IDONE = 0
      JNUM = 0
      DO 180 I = 1,NPROD
      JDUM = JPROD1(I)
      IF(JDUM.GT.0) GO TO 167
      IDONE = IDONE+1
      GO TO 180
167   IF (ITIME .LT. JOB2(JDUM) ) GO TO 180
      COST1 = RJOB(JDUM)
      COST2 = COST1
      INUM = 1
169   IF(COST.GE.COST2) GO TO 170
      COST = COST2
      JNUM = JDUM
```

```
170    IF (JDUM .EQ. JPROD2(JOB1(JDUM) ) ) GO TO 180
       JDUM = JOB4(JDUM)
       COST1 = COST1+PJOB(JDUM)
       INUM = INUM+1
       COST2 = COST1/INUM
       GO TO 159
180    CONTINUE
       IF(JNUM.GT.0) GO TO 184
       IF(IDONE.EQ.NPROD) GO TO 192
       ITIME = ITIME+1
       GO TO 191
C***********************************************************************
C*     PLACE JOB JNUM'S SUBSTRING IN THE JOB SCHEDULE, WITH JDUM AS THE *
C*     FIRST JOB ON THE SUBSTRING. THEN UPDATE THE SCHEDULE COST FOR    *
C*     THESE JOBS.                                                      *
C***********************************************************************
184    JDUM = JPROD1( JOB1(JNUM) )
185    SCHED(MACH(ITIME) - NMACH(ITIME) + 1, ITIME, 2) = JDUM
       COPT = COPT + (ITIME - JOB2(JDUM)) * RJOB(JDUM)
       NMACH(ITIME) = NMACH(ITIME)-1
       IF(NMACH(ITIME).GT.0) GO TO 189
       ITIME = ITIME+1
       IF(ITIME.GT.TIME) GO TO 192
189    IF(JDUM.EQ.JNUM) GO TO 190
       JDUM = JOB4(JDUM)
       GO TO 185
190    JPROD1( JOB1(JNUM) ) = JOB4(JNUM)
191    IF(ITIME.LE.TIME) GO TO 155
C***********************************************************************
C*     THE SCHEDULE HAS BEEN COMPLETED. ATTEMPT TO IMPROVE IT VIA PAIR- *
C*     WISE JOB INTERCHANGES. THEN RETURN                               *
C***********************************************************************
192    IF(FLAG) CALL PAIR(2,COPT,FLAG1,FLAG2)
       IF(COPT.LT.HCOST) HCOST = COPT
       RETURN
       END
```

```
      SUBROUTINE PAIR(K,COST,FLAG1,FLAG2)
C**********************************************************************
C**********************************************************************
C*    SUBROUTINE PAIR
C*    AN ITERATIVE PROCEDURE TO IMPROVE A JOB SCHEDULE USING PAIRWISE  *
C*    INTERCHANGES OF JOBS. NOTE IT IS ASSUMED THAT AT LEAST ONE JOB IS*
C*    AVAILABLE DURING THE FIRST TIME PERIOD.                          *
C**********************************************************************
C**********************************************************************
      IMPLICIT INTEGER*4 (A-Z)
      COMMON LPRD(1000),TIPRD(1000),JIPROD(20),JPROD1(20),JPROD2(20),
     *       MACH(200),JMACH(200),NMACH(200),RJOB(200),SJOB(200),
     *       JOB1(200),JOB3(200),JOB2(200),JOB4(200),DJOB(200),
     *       SCHED(5,20,?),TIME,NJOB,NPROD,NCOST,COPT,NSCHED,TIME1,
     *       TIME2,PRINT
      LOGICAL PRINT, FLAG1,FLAG2
C**********************************************************************
C*    INITIALIZE IMPORTANT VECTORS
C**********************************************************************
      IF (K .EQ. 3) GO TO 24
      DO 2 I = 1,NJOB
      SJOB(I) = 0
2     CONTINUE
      DO 4 J = 1, TIME
      MACHJ = MACH(J)
      DO 3 I = 1, MACHJ
      IJK = SCHED(I,J,K)
      IF (IJK .EQ. 0) GO TO 3
      SJOB(IJK) = J
3     CONTINUE
4     CONTINUE
C**********************************************************************
C*    DECOMPOSE THE TIME HORIZON INTO SECTIONS WITH NO UNUSED MACHINES *
C*    IN ANY SECTION. NOTE SJOB INDICATES THE TIME PERIOD A JOB IS     *
C*    SCHEDULED FOR.                                                   *
C**********************************************************************
      TIME1 = 1
5     DO 10 I = TIME1, TIME
      IF ( SCHED(MACH(I), I, K) .EQ. 0 ) GO TO 11
10    CONTINUE
      TIME2 = TIME
      GO TO 14
11    IF ( SCHED(1,I,K) .EQ. 0) GO TO 13
      TIME2 = I
      GO TO 14
13    TIME2 = I - 1
14    IF (TIME2 .EQ. TIME1) GO TO 503
C**********************************************************************
C*    DETERMINE IF THERE IS A JOB WHICH MAY BE DELAYED IN THE JOB     *
C*    SCHEDULE.                                                       *
C**********************************************************************
24    T1 = TIME2 - 1
25    DO 100 J = TIME1, T1
      MACHJ = MACH(J)
      DO 95 I = 1, MACHJ
      JOB1 = SCHED(I,J,K)
      IF (JOB1 .EQ. 0) GO TO 500
      SUCC1 = JOB4(JOB1)
      IF (SUCC1 .EQ. 0) GO TO 35
      LIMIT = SJOB(SUCC1)
      IF (LIMIT .EQ. 0) GO TO 35
      IF (LIMIT .EQ. J) GO TO 95
      GO TO 38
C**********************************************************************
C*    DETERMINE IF THERE IS A JOB IN A LATER TIME PERIOD WITH A HIGHER *
C*    DEFERRAL COST THAN JOB1 AND WHICH CAN BE EXCHANGED WITH JOB1     *
C*    WITHOUT VIOLATING ORDERING CONSTRAINTS.                          *
```

```
C*************************************************************************
35     LIMIT = TIME2
38     NEXT = J + 1
C
       DO 90 J1 = NEXT, LIMIT
       MACHJ1 = MACH(J1)
       DO 85 I1 = 1, MACHJ1
       JOBI1 = SCHED(I1,J1,K)
       IF (JOBI1 .EQ. 0) GO TO 95
       IF (RJOB(JOBI) .GE. RJOB(JOBI1) ) GO TO 85
       IF (JOB2(JOBI1) .GT. J) GO TO 85
       IF(JOB1(JOBI) .EQ. JOB1(JOBI1)) GO TO 85
       PREC2 = JOB3(JOBI1)
       IF (PREC2 .EQ. 0) GO TO 125
       IF (SJOB(PREC2) .GT. J) GO TO 85
       GO TO 125
85     CONTINUE
90     CONTINUE
95     CONTINUE
100    CONTINUE
       GO TO 500
C*************************************************************************
C*     INTERCHANGE JOBI AND JOB2. UPDATE THE SJOB VECTOR AND THE         *
C*     SCHEDULE COST. THEN BEGIN THE PROCEDURE AGAIN WITH THE REVISED    *
C*     SCHEDULE.                                                         *
C*************************************************************************
125    SCHED(I,J,K) = JOBI1
       SCHED(I1,J1,K) = JOBI
       SJOB(JOBI) = J1
       SJOB(JOBI1) = J
       COST = COST - (J1 - J) * (RJOB(JOBI1) - RJOB(JOBI) )
       GO TO 25
C*************************************************************************
C*     THE JOB SCHEDULE FOR THIS TIME SECTION CANNOT BE IMPROVED BY      *
C*     PAIRWISE INTERCHANGES. ATTEMPT TO IMPROVE THE JOB SCHEDULE OF     *
C*     THE SUCCEEDING TIME SECTION. IF ALL SECTIONS HAVE BEEN COMPLETED *
C*     RETURN.                                                           *
C*************************************************************************
500    CHECK = COST
       IF(.NOT.FLAG1) GO TO 503
       CALL CHANGE(K,COST)
       IF(CHECK.GT.COST) GO TO 24
       IF(.NOT.FLAG2) GO TO 503
       CALL WAY4(K,COST)
       IF (CHECK .GT. COST) GO TO 24
503    IF (TIME2 .EQ. TIME) RETURN
       IF (K .EQ. 3) RETURN
       T = TIME2 + 1
       T1 = TIME - 1
       DO 505 TIME1 = T, T1
505    IF (SCHED(1, TIME1, K) .GT. 0) GO TO 5
       RETURN
       END
```

```
      SUBROUTINE CHANGE(K,COST)
C**************************************************************************
C**************************************************************************
C*     SUBROUTINE CHANGE
C*     THIS PROCEDURE ATTEMPTS TO FIND A THREE JOB PIVOT WHICH WILL       *
C*     REDUCE THE SCHEDULE COST.                                          *
C**************************************************************************
C**************************************************************************
      IMPLICIT INTEGER*4 (A-Z)
      COMMON LEND(1000),TLEND(1000),J1PROD(20),JPROD1(20),JPROD2(20),
     *       MACH(200),JMACH(200),NMACH(200),RJOB(200),SJOB(200),
     *       JOB1(200),JOB2(200),JOB3(200),JOB4(200),DJOB(200),
     *       SCHED(5,200,3),TIME,NJOB,NPROD,HCOST,COPT,MSCHED,TIME1,
     *       TIME2,PRINT
      LOGICAL PRINT
C**************************************************************************
C*     DETERMINE IF THERE IS A JOB PAIR WHICH MAY BE DELAYED IN THE JOB  *
C*     SCHEDULE.                                                          *
C**************************************************************************
      T2 = TIME2 - 2
      IF (T2 .LT. TIME1) RETURN
      DO 100 J = TIME1, T2
      MACHJ = MACH(J)
      DO 95 I = 1, MACHJ
      JOBI = SCHED(I,J,K)
      IF (JOBI .EQ. 0) GO TO 200
      SUCC1 = JOB4(JOBI)
      LIMIT = TIME2 - 1
      IF (SUCC1 .EQ. 0) GO TO 30
      SUCCT = SJOB(SUCC1)
      IF (SUCCT .EQ. 0) GO TO 30
      IF (SUCCT .EQ. J) GO TO 95
      IF (SUCCT .LT. LIMIT) LIMIT = SUCCT
30    J1 = J + 1
      DO 80 J2 = J1, LIMIT
      MACHJ2 = MACH(J2)
      DO 75 I2 = 1, MACHJ2
      JOBII = SCHED(I2,J2,K)
      IF (JOBII .EQ. 0) GO TO 95
      IF (RJOB(JOBI) .GE. RJOB(JOBII)) GO TO 75
      LIMIT2 = TIME2
      SUCC2 = JOB4(JOBII)
      IF (SUCC2 .EQ. 0) GO TO 35
      SUCCT = SJOB(SUCC2)
      IF (SUCCT .EQ. 0) GO TO 35
      IF(SUCCT .EQ. J2) GO TO 75
      IF (SUCCT .LT. LIMIT2) LIMIT2 = SUCCT
C**************************************************************************
C*     DETERMINE IF THERE IS A JOB IN A LATER TIME PERIOD WHICH MAY BE   *
C*     PIVOTED WITH JOBS JOBI AND JOBII TO REDUCE THE SCHEDULE COST.      *
C**************************************************************************
35    NEXT = J2 + 1
      DO 60 J3 = NEXT, LIMIT2
      MACHJ3 = MACH(J3)
      DO 55 I3 = 1, MACHJ3
      JOBIII = SCHED(I3,J3,K)
      IF (JOBIII .EQ. 0) GO TO 75
      TYPE = JOB1(JOBIII)
      IF (TYPE .EQ. JOB1(JOBI) ) GO TO 55
      IF (TYPE .EQ. JOB1(JOBII) ) GO TO 55
      DIFF = RJOB(JOBI) * (J2 - J) + RJOB(JOBII) * (J3 - J2)
     *       - RJOB(JOBIII) * (J3 - J)
      IF (DIFF .GE. 0) GO TO 55
      IF ( JOB2(JOBIII) .GT. J) GO TO 55
      PREC3 = JOB3(JOBIII)
      IF (PREC3 .EQ. 0) GO TO 125
      IF (SJOB(PREC3) .GT. J) GO TO 55
```

```
      GO TO 125
55    CONTINUE
60    CONTINUE
75    CONTINUE
80    CONTINUE
95    CONTINUE
100   CONTINUE
      GO TO 200
C**********************************************************************
C*    PIVOT THE JOB GROUP'S SCHEDULE TIMES, UPDATE THE SJOB VECTOR AND *
C*    THE SCHEDULE COST, THEN RETURN TO PAIR.                          *
C**********************************************************************
125   SCHED(I,J,K) = JOBIII
      SCHED(I2,J2,K) = JOBI
      SCHED(I3,J3,K) = JOBII
      SJOB(JOBIII) = J
      SJOB(JOBI) = J2
      SJOB(JOBII) = J3
      COST = COST + DIFF
      RETURN
C**********************************************************************
C*    DETERMINE IF THERE IS A JOB PAIR WHICH SHOULD BE CONSIDERED FOR  *
C*    ADVANCEMENT IN THE JOB SCHEDULE.                                 *
C**********************************************************************
200   T1 = TIME1 + 1
      T2 = TIME2 - 1
      DO 300 J = T1, T2
      MACHJ = MACH(J)
      DO 295 I = 1, MACHJ
      JOBI = SCHED(I,J,K)
      IF (JOBI .EQ. 0) GO TO 500
      LIMIT = TIME1
      PRECI = JOB3(JOBI)
      IF (PRECI .EQ. 0) GO TO 225
      PRECT = SJOB(PRECI)
      IF (PRECT .GT. LIMIT) LIMIT = PRECT
225   AVAILI = JOB2(JOBI)
      IF (AVAILI .GT. LIMIT) LIMIT = AVAILI
      IF (LIMIT .EQ. J) GO TO 295
      J1 = J + 1
      DO 285 J2 = J1, TIME2
      MACHJ2 = MACH(J2)
      DO 280 I2 = 1, MACHJ2
      JOBII = SCHED(I2,J2,K)
      IF (JOBII .EQ. 0) GO TO 295
      IF (RJOB(JOBI) .GE. RJOB(JOBII) ) GO TO 280
      IF (JOB2(JOBII) .GT. J) GO TO 290
      PREC2 = JOB3(JOBII)
      IF (PREC2 .EQ. 0) GO TO 230
      IF (SJOB(PREC2) .GT. J) GO TO 260
C**********************************************************************
C*    DETERMINE IF THERE IS A JOB IN AN EARLIER TIME PERIOD WHICH MAY  *
C*    BE PIVOTED WITH JOBS JOBI AND JOBII TO REDUCE THE SCHEDULE COST. *
C**********************************************************************
230   J0 = J - 1
      DO 270 J3 = LIMIT, J0
      MACHJ3 = MACH(J3)
      DO 265 I3 = 1, MACHJ3
      JOBIII = SCHED(I3,J3,K)
      IF (JOBIII .EQ. 0) GO TO 280
      TYPE = JOB1(JOBIII)
      IF (TYPE .EQ. JOB1(JOBI) ) GO TO 265
      IF (TYPE .EQ. JOB1(JOBII) ) GO TO 265
      DIFF = RJOB(JOBIII) * (J2 - J3) - RJOB(JOBI) * (J - J3)
     *       - RJOB(JOBII) * (J2 - J)
      IF (DIFF .GE. 0) GO TO 265
      SUCC3 = JOB4(JOBIII)
      IF (SUCC3 .EQ. 0) GO TO 325
      IF ( SJOB(SUCC3) .EQ. 0) GO TO 325
      IF (SJOB(SUCC3) .LT. J2) GO TO 265
      GO TO 325
265   CONTINUE
270   CONTINUE
280   CONTINUE
285   CONTINUE
295   CONTINUE
300   CONTINUE
      RETURN
C**********************************************************************
C*    PIVOT THE JOB GROUP'S SCHEDULE TIMES, UPDATE THE SJOB VECTOR AND *
C*    THE SCHEDULE COST.                                               *
C**********************************************************************
325   SCHED(I,J,K) = JOBII
      SCHED(I2,J2,K) = JOBIII
      SCHED(I3,J3,K) = JOBI
      SJOB(JOBII) = J
      SJOB(JOBIII) = J2
      SJOB(JOBI) = J3
      COST = COST + DIFF
500   RETURN
      END
```

```
      SUBROUTINE WAY4(K,COST)
C*********************************************************************
C*   THIS IS A 4-WAY INTERCHANGE METHOD WHICH ASSUMES ALL
C*   2-WAY AND 3-WAY INTERCHANGES HAVE BEEN MADE.  THERE ARE 3 SECTIONS
C*   TO THIS SUBROUTINE. IN ALL, JOBI THRU JOBIV ARE ORDERED BY POSITION
C*   IN THE INPUT SCHEDULE.  JOBI IS FIRST.  IN THIS SECTION JOBI AND
C*   JOBIII ARE INTERCHANGED AS ARE JOBII AND JOBIV.
C*********************************************************************
      IMPLICIT INTEGER*4 (A-Z)
      REAL Z,TTIME,CLIMIT, POSS
      COMMON LRND(1000),ICRNO(1000),JIPROD(20),JPROD1(20),JPROD2(20),
     *       MACH(200),JMACH(200),NMACH(200),RJOB(200),SJOB(200),
     *       JOB1(200),JOB2(200),JOB3(200),JOB4(200),DJOB(200),
     *       SCHED(5,200,3),TIME,NJOB,NPROD,HCOST,COPY,MCHED,TIME1,
     *       TIME2,PRINT
      DAT1 = TIME1+1
      DO 135 DJ4 = DAT1,TIME2
      J4 = TIME2-(DJ4+DAT1)
      MACHJ4 = MACH(J4)
      DO 130 M4 = 1,MACHJ4
      JOBIV = SCHED(M4,J4,K)
      IF(JOBIV.EQ.0) GO TO 130
      D2T1 = JOB2(JOBIV)
      IF(D2T1.EQ.J4) GO TO 130
      D2T2 = J4-1
      DO 125 J2 = D2T1,D2T2
      MACHJ2 = MACH(J2)
      DO 120 M2 = 1,MACHJ2
      JOBII = SCHED(M2,J2,K)
      IF(JOBII.EQ.0) GO TO 125
      IF(JOB1(JOBII).EQ.JOB1(JOBIV)) GO TO 120
      DUM = JOB4(JOBII)
      IF(DUM.EQ.0) GO TO 50
      IF(SJOB(DUM).LT.J4) GO TO 120
50    RJ24 = RJOB(JOBIV)-RJOB(JOBII)
      IF(RJ24.GE.0) GO TO 80
      JOBI = JOB2(JOBII)
      IF(JOBI.EQ.0) GO TO 120
      J1 = SJOB(JOBI)
      M1 = 0
55    M1 = M1+1
      IF(JOBI.NE.SCHED(M1,J1,K)) GO TO 55
      D3T1 = J2+1
      DO 75 J3 = D3T1,J4
      MACHJ3 = MACH(J3)
      DO 70 M3 = 1,MACHJ3
      JOBIII = SCHED(M3,J3,K)
      IF(JOBIII.EQ.0) GO TO 75
      IF(JOBIII.EQ.JOBIV) GO TO 70
      IF(JOB1(JOBIII).EQ.JOB1(JOBI)) GO TO 70
      IF(JOB2(JOBIII).GT.J1) GO TO 70
      DUM = JOB3(JOBIII)
      IF(DUM.EQ.0) GO TO 60
      IF(SJOB(DUM).GT.J1) GO TO 70
60    DUM = JOB3(JOBIV)
      IF(DUM.EQ.0) GO TO 65
      IF(DUM.EQ.JOBIII) GO TO 65
      IF(SJOB(DUM).GT.J2) GO TO 70
65    DIFF = RJ24*(J2-J4)+(RJOB(JOBIII)-RJOB(JOBI))*(J1-J3)
      IF(DIFF.LT.0) GO TO 140
70    CONTINUE
75    CONTINUE
      GO TO 120
80    JOBIII = JOB3(JOBIV)
      IF(JOBIII.EQ.0) GO TO 120
      D1T1 = JOB3(JOBIII)
      IF(D1T1.EQ.0) GO TO 85
```

```
      DIT1 = SJOB(DIT1)
85    DUM = JOB2(JOBIII)
      IF(DIT1.LT.DUM) DIT1 = DUM
      IF(DIT1.GT.J2) GO TO 120
      J3 = SJOB(JOBIII)
      IF(J3.LT.J2+1) GO TO 120
      M3 = 0
90    M3 = M3+1
      IF(JOBIII.NE.SCHED(M3,J3,K)) GO TO 90
      J1 = DIT1-1
95    J1 = J1+1
      MACHJ1 = MACH(J1)
      M1 = 0
100   M1 = M1+1
      JOBI = SCHED(M1,J1,K)
      IF(JOBI.EQ.0) GO TO 110
      IF(JOB1(JOBI).EQ.JOB1(JOBIV)) GO TO 110
      IF(JOBI.EQ.JOBII) GO TO 110
      DUM = JOB4(JOBI)
      IF(DUM.EQ.0) GO TO 105
      IF(DUM.EQ.JOBII) GO TO 105
      IF(SJOB(DUM).LT.J3) GO TO 110
105   DIFF = RJ2*(J2-J4)+(RJOB(JOBIII)-RJOB(JOBI))*(J1-J3)
      IF(DIFF.LT.0) GO TO 140
110   IF(M1.LT.MACHJ1) GO TO 100
115   IF(J1.LT.J2) GO TO 95
120   CONTINUE
125   CONTINUE
130   CONTINUE
135   CONTINUE
      GO TO 145
140   SCHED(M1,J1,K) = JOBIII
      SCHED(M2,J2,K) = JOBIV
      SCHED(M3,J3,K) = JOBI
      SCHED(M4,J4,K) = JOBII
      SJOB(JOBI) = J3
      SJOB(JOBII) = J4
      SJOB(JOBIII) = J1
      SJOB(JOBIV) = J2
      COST = COST+DIFF
      RETURN
C*************************************************************************
C*   IN THE SECOND SECTION JOBI THRU JOBIV ARE ALL IN DIFFERENT PERIODS
C*   JOBIV REPLACES JOBI WHICH REPLACES JOBII.   JOBII REPLACES JOBIII
C*   WHICH REPLACES JOBIV.
C*************************************************************************
145   IF(TIME2-TIME1.LT.4) RETURN
      DAT1 = TIME1+3
      DO 205 DJ4 = DAT1, TIME2
      J4 = TIME2-DJ4+DAT1
      MACHJ4 = MACH(J4)
      DO 200 M4 = 1,MACHJ4
      JOBIV = SCHED(M4,J4,K)
      IF(JOBIV.EQ.0) GO TO 200
      DIT1 = JOB3(JOBIV)
      IF(DIT1.EQ.0) GO TO 150
      DIT1 = SJOB(DIT1)
150   DUM = JOB2(JOBIV)
      IF(DIT1.LT.DUM) DIT1 = DUM
      DIT2 = J4-3
      IF(DIT1.GT.DIT2) GO TO 200
      DO 195 J1 = DIT1,DIT2
      MACHJ1 = MACH(J1)
      DO 190 M1 = 1,MACHJ1
      JOBI = SCHED(M1,J1,K)
      IF(JOBI.EQ.0) GO TO 190
      D2T1 = J1+1
      D2T2 = J4-2
      DUM = JOB4(JOBI)
      IF(DUM.EQ.0) GO TO 155
      DUM = SJOB(DUM)
      IF(DUM.LT.D2T2) D2T2=DUM
```

```
      IF(D2T1.GT.D2T2) GO TO 190
155   IF(JOB1(JOBI).EQ.JOB1(JOBIV)) GO TO 190
      IF(RJOB(JOBIV).LE.RJOB(JOBI)) GO TO 190
      DO 185 J2 = D2T1,D2T2
      MACHJ2 = MACH(J2)
      DO 180 M2 = 1,MACHJ2
      JOBII = SCHED(M2,J2,K)
      IF(JOBII.EQ.0) GO TO 180
      D3T1 = J2+1
      D3T2 = J4-1
      DUM = JOB4(JOBII)
      IF(DUM.EQ.0) GO TO 160
      DUM = SJOB(DUM)
      IF(DUM.LT.D3T2) D3T2=DUM
      IF(D3T1.GT.D3T2) GO TO 180
160   DO 175 J3 = D3T1,D3T2
      MACHJ3 = MACH(J3)
      DO 170 M3 = 1,MACHJ3
      JOBIII = SCHED(M3,J3,K)
      IF(JOBIII.EQ.0) GO TO 170
      DUM = JOB4(JOBIII)
      IF(DUM.EQ.0) GO TO 165
      IF(J4.GT.SJOB(DUM)) GO TO 170
165   DIFF = RJOB(JOBIV)*(J1-J4)+RJOB(JOBI)*(J2-J1)+RJOB(JOBII)
     *       *(J3-J2)+RJOB(JOBIII)*(J4-J3)
      IF(DIFF.LT.0) GO TO 210
170   CONTINUE
175   CONTINUE
180   CONTINUE
185   CONTINUE
190   CONTINUE
195   CONTINUE
200   CONTINUE
205   CONTINUE
      GO TO 215
210   SCHED(M1,J1,K) = JOBIV
      SCHED(M2,J2,K) = JOBI
      SCHED(M3,J3,K) = JOBII
      SCHED(M4,J4,K) = JOBIII
      SJOB(JOBI) = J2
      SJOB(JOBII) = J3
      SJOB(JOBIII) = J4
      SJOB(JOBIV) = J1
      COST = COST+DIFF
      RETURN
C*****************************************************************************
C*   IN THE THIRD SECTION JOBI THRU JOBIV ARE AGAIN ALL IN DIFFERENT
C*   PERIODS.  JOBI REPLACES JOBIV WHICH REPLACES JOBIII.  JOBIII
C*   REPLACES JOBII WHICH REPLACES JOBI.
C*****************************************************************************
215   D1T2 = TIME2-3
      DO 275 J1 = TIME1,D1T2
      MACHJ1 = MACH(J1)
      DO 270 M1 = 1,MACHJ1
      JOBI = SCHED(M1,J1,K)
      IF(JOBI.EQ.0) GO TO 270
      D4T2 = JOB4(JOBI)
      IF(D4T2.EQ.0) GO TO 220
      D4T2 = SJOB(D4T2)
220   IF(D4T2.GT.TIME2) D4T2=TIME2
      D4T1 = J1+3
      IF(D4T1.GT.D4T2) GO TO 270
      DO 265 DJ4 = D4T1,D4T2
      J4 = D4T2-DJ4+D4T1
      MACHJ4 = MACH(J4)
      DO 260 M4 = 1,MACHJ4
      JOBIV = SCHED(M4,J4,K)
      IF(JOBIV.EQ.0) GO TO 260
      IF(RJOB(JOBIV).LE.RJOB(JOBI)) GO TO 260
      IF(JOB1(JOBIV).EQ.JOB1(JOBI)) GO TO 260
      D3T2 = J4-1
      D3T1 = JOB3(JOBIV)
```

```
      IF(D3T1.EQ.0) GO TO 225
      D3T1 = SJOB(D3T1)
225   DUM = JOB2(JOBIV)
      IF(D3T1.LT.DUM) D3T1=DUM
      DUM = J1+2
      IF(D3T1.LT.DUM) D3T1=DUM
      IF(D3T1.GT.D3T2) GO TO 260
      DO 255 DJ3=D3T1,D3T2
      J3 = D3T2-DJ3+D3T1
      MACHJ3 = MACH(J3)
      DO 250 M3 = 1,MACHJ3
      JOBIII = SCHED(M3,J3,K)
      IF(JOBIII.EQ.0) GO TO 250
      D2T2 = J3-1
      D2T1 = JOB3(JOBIII)
      IF(D2T1.EQ.0) GO TO 230
      D2T1 = SJOB(D2T1)
230   DUM = JOB2(JOBIII)
      IF(D2T1.LT.DUM) D2T1=DUM
      DUM=J1+1
      IF(D2T1.LT.DUM) D2T1=DUM
      IF(D2T1.GT.D2T2) GO TO 250
      DO 245 DJ2 = D2T1,D2T2
      J2 = D2T2-DJ2+D2T1
      MACHJ2 = MACH(J2)
      DO 240 M2 = 1,MACHJ2
      JOBII = SCHED(M2,J2,K)
      IF(JOBII.EQ.0) GO TO 240
      DUM = JOB3(JOBII)
      IF(DUM.EQ.0) GO TO 235
      DUM = SJOB(DUM)
235   D1T1 = JOB2(JOBII)
      IF(DUM.LT.D1T1) DUM=D1T1
      IF(DUM.GT.J1) GO TO 240
      DIFF = RJOB(JOBI)*(J4-J1)+RJOB(JOBII)*(J1-J2)+RJOB(JOBIII)
     *       *(J2-J3)+RJOB(JOBIV)*(J3-J4)
      IF(DIFF.LT.0) GO TO 280
240   CONTINUE
245   CONTINUE
250   CONTINUE
255   CONTINUE
260   CONTINUE
265   CONTINUE
270   CONTINUE
275   CONTINUE
      RETURN
280   SCHED(M1,J1,K) = JOBII
      SCHED(M2,J2,K) = JOBIII
      SCHED(M3,J3,K) = JOBIV
      SCHED(M4,J4,K) = JOBI
      SJOB(JOBI) = J4
      SJOB(JOBII) = J1
      SJOB(JOBIII) = J2
      SJOB(JOBIV) = J3
      COST = COST+DIFF
      RETURN
      END
```

BIBLIOGRAPHY

1. Adolphson, D. and Hu, T. C., "Optimal Linear Ordering," S.I.A.M. J. Appl. Math., Vol. 25 (1973), 403-423.

2. Bartholdi, J. J., Martin-Vega, L., and Ratliff, H. D., "Efficient Network Solutions to Parallel Processor Scheduling Problems: A Survey," University of Florida, Industrial and Systems Engineering Department, Research Report 76-16, Gainesville, FL (1976).

3. Berman, E. B. and Clark, A. J., "An Optimal Inventory Policy for a Military Organization," The Rand Corporation, P-647, Santa Monica, CA (1955).

4. Bessler, S. A. and Veinott, A. G., Jr., "Optimal Policy for a Dynamic Multi-Echelon Inventory Model," Nav. Res. Log. Quart., Vol. 13 (1966), 355-389.

5. Clark, A. J., "A Dynamic, Single-Item, Multi-Echelon Inventory Model," The Rand Corporation, RM-2297, Santa Monica, CA (1958).

6. Clark, A. J., "An Informal Survey of Multi-Echelon Inventory Theory," Nav. Res. Log. Quart., Vol. 19 (1972), 621-650.

7. Clark, A. J. and Scarf, H., "Approximate Solutions to a Simple Multi-Echelon Inventory Problem," Chap. 5 in K. J. Arrow, S. Karlin, and H. Scarf (Eds.): Studies in Applied Probability and Management Science, (Stanford University Press, Stanford, CA, 1962).

8. Clark, A. J. and Scarf, H., "Optimal Policies for a Multi-Echelon Inventory Problem,"Management Science, Vol. 6 (1960), 475-490.

9. Connors, M. M. and Zangwill, W. I., "Cost Minimization in Networks with Discrete Stochastic Requirements," Operations Research, Vol. 19 (1971), 794-821.

10. Crowston, W. B. and Wagner, M. H., "Dynamic Lot Size Models for Multi-Stage Assembly Systems," Management Science, Vol. 20 (1973), 14-21.

11. Crowston, W. B., Wagner, M. and Williams, J., "Economic Lot Size Determination in Multi-Stage Assembly Systems," Management Science, Vol. 19 (1973), 517-527.

12. Dorsey, R. C., Hodgson, T. J., and Ratliff, H. D., "A Network Approach to a Multi-Facility, Multi-Product Production Scheduling Problem without Backordering," Management Science, Vol. 21 (1975), 813-822.

13. Elmaghraby, S. E. and Sarin, S. C., "On Scheduling Precedence-Related Jobs on Parallel Machines: Bounds on the Performance of a Heuristic," North Carolina State University, Research Report No. 117, Raleigh, NC (1977).

14. Evans, G. W. H., "A Transportation and Production Model," Nav. Res. Log. Quart., Vol. 5 (1958), 137-154.

15. Fukuda, Y., "Bayes and Maximum Likelihood Policies for a Multi-Echelon Inventory Problem," Planning Research Corporation, R-161, Los Angeles, CA (1960).

16. Fukuda, Y., "Optimal Disposal Policies," Nav. Res. Log. Quart., Vol. 8 (1961), 221-227.

17. Gross, D., "Centralized Inventory Control in Multilocation Supply Systems," Chap. 3 in H. Scarf, D. Gilford, and M. Shelly (Eds.): Multi-Stage Inventory Models and Techniques, (Stanford University Press, Stanford, CA, 1963).

18. Hadley, G. and Whitin, T. M., "A Model for Procurement, Allocation and Redistribution for Low Demand Items," Nav. Res. Log. Quart., Vol. 8 (1961), 395-414.

19. Hadley, G. and Whitin, T. M., "An Inventory Transportation Model with N Locations," Chap. 5 in H. Scarf, D. Gilford, and M. Shelly (Eds.): Multi-Stage Inventory Models and Techniques, (Stanford University Press, Stanford, CA, 1963).

20. Hochstaedter, D., "An Approximation of the Cost Function for Multi-Echelon Inventory Model," Management Science, Vol. 16 (1970), 716-727.

21. Hodgson, T. J. and Loveland, C. S., "A Partial LaGrange Multiplier Approach to a Resource Constrained C.P.M. Problem," University of Florida, Industrial and Systems Engineering Department, Research Report 76-11, Gainesville, FL (1976).

22. Hodgson, T. J. and Loveland, C. S., "An Analytic Approach for a Capacitated C.P.M. Problem," University of Florida, Industrial and Systems Engineering Department, Research Report 76-12, Gainesville, FL (1976).

23. Horn, W. A., "Single-Machine Job Sequencing with Treelike Precedence Ordering and Linear Delay Penalties," S.I.A.M. J. Appl. Math., Vol. 23 (1972), 189-202.

24. Ignall, E. and Veinott, A. F., Jr., "Optimality of Myopic Inventory Policies for Several Substitute Products," Management Science, Vol. 15 (1969), 284-304.

25. Jensen, P. A. and Khan, H. A., "Scheduling a Multi-Stage Production System with Set-Up and Inventory Costs," AIIE Transactions, Vol. 4 (1972), 126-133.

26. Johnson, L. A. and Montgomery, D. C., Operations Research in Production Planning, Scheduling and Inventory Control, (John Wiley & Sons, New York, 1974).

27. Kalymon, B. A., "A Decomposition Algorithm for Arborescence Inventory Systems," Operations Research, Vol. 20 (1972), 860-874.

28. Krishnan, K. S. and Rao, V. R. K., "Inventory Control in Warehouses," J. Indust. Eng., Vol. 16 (1965), 212-215.

29. Lawler, E. L., "Optimal Sequencing of Jobs Subject to Series Parallel Precedence Constraints," The Mathematical Centre, Amsterdam, Netherlands (1975).

30. Lenstra, J. K., "Sequencing by Enumerative Methods," The Mathematical Centre, Amsterdam, Netherlands (1976).

31. Love, R. F., "A Two-Station Stochastic Inventory Model with Exact Methods of Computing Optimal Policies," Nav. Res. Log. Quart., Vol. 14 (1967), 185-217.

32. Love, Stephen F., "A Facilities in Series Inventory Model with Nested Schedules," Management Science, Vol. 18 (1972), 327-338.

33. Ratliff, H. D., "Networks Models for Production Scheduling Problems with Convex Cost and Batch Processing," University of Florida, Industrial and Systems Engineering Department, Research Report 76-18, Gainesville, Fla. (1976).

34. Rosenman, B. and Hockstra, D., "A Management System for High-Value Army Aviation Components," U. S. Army, Advanced Logistics Research Office, Frankfort Arsenal, Report No. TR64-1, Philadelphia, Pa. (1964).

35. Schwarz, L. B. and Schrage, L., "Optimal and Systems Myopic Policies for Multi-Echelon on Production/Inventory Assembly Systems," Management Science, Vol. 21 (1975), 1285-1294.

36. Sherbrooke, C. C., "Metric: A Multi-Echelon Technique for Recoverable Item Control," Operations Research, Vol. 16 (1968), 122-141.

37. Sidney, J. B., "Decomposition Algorithms for Single Machine Sequencing with Precedence Relations and Deferral Costs," Operations Research, Vol. 23 (1975), 283-298.

38. Simon, R. M., "Stationary Properties of a Two-Echelon Inventory Model for Low Demand Items," Operations Research, Vol. 19, (1971), 761-773.

39. Sobel, M. J., "Smoothing Start-Up and Shut-Down Costs in Sequential Production," Operations Research, Vol. 17 (1969), 133-144.

40. Szendrovits, A. Z., "Manufacturing Cycle Time Determination for a Multi-Stage Economic Production Quantity Model," Management Science, Vol. 22 (1975), 298-308.

41. Taha, H. A. and Skeith, R. W., "The Economic Lot Sizes in Multi-Stage Production Systems," A.I.I.E. Transactions, Vol. 2 (1970), 157-162.

42. Thomas, A. B., "Optimizing a Multi-Stage Production Process," Operational Research Quarterly, Vol. 14 (1963), 201-213.

43. Veinott, A. F., Jr., "Minimum Concave Cost Solution of Leontief Substitution Models of Multi-Facility Inventory Systems," Operations Research, Vol. 17 (1969), 262-291.

44. Von Lanzenauer, C. H., "A Production Scheduling Model by Bivalent Linear Programming," Management Science, Vol. 17 (1970), 105-111.

45. Williams, J. F., "Multi-Echelon Production Scheduling When Demand Is Stochastic," University of Wisconsin, School of Business Administration, Milwaukee, Wisc. (1971).

46. Young, H. H., "Optimization Models for Production Lines," Journal of Industrial Engineering, Vol. 18 (1967), 70-78.

47. Zacks, S., "A Two-Echelon, Multi-Station Inventory Model for Navy Applications," Nav. Res. Log. Quart., Vol. 17 (1970), 79-85.

48. Zacks, S., "Bayes Adaptive Control of Two-Echelon Multi-Station Inventory Systems," The George Washington University, Institute for Management Science and Engineering, TN-61541, Washington, D.C. (1970).

49. Zangwill, W., "A Backlogging Model and a Multi-Echelon Model of a Dynamic Economic Lot Size Production System - A Network Approach," Management Science, Vol. 15 (1969), 506-527.

50. Zangwill, W., "A Deterministic Multi-Product, Multi-Facility Production and Inventory System," Operations Research, Vol. 14 (1966), 486-508.

BIOGRAPHICAL SKETCH

Charles Stafford Loveland was born on August 22, 1945, in La Crosse, Wisconsin. He received his elementary education at Emerson Orthopedic School and graduated from Aquinas High School in La Crosse in 1963. He attended Southern Illinois University in Carbondale, Illinois, from 1963 to 1970, receiving a bachelor's degree in Mathematics (1968) and a master's degree in Computer Science (1970). In 1970, he entered the graduate program in the Industrial and Systems Engineering Department at the University of Florida, receiving a master's degree in Operations Research (1975).

Charles Stafford Loveland is a member of Tau Beta Pi and Alpha Pi Mu honorary engineering societies, as well as The Institute of Management Science and Operations Research Society of America professional societies.

I certify that I have read this study and that in my opinion it conforms to acceptable standards of scholarly presentation and is fully adequate, in scope and quality, as a dissertation for the degree of Doctor of Philosophy.

Dr. Thom J. Hodgson, Chairman
Professor of Industrial and
Systems Engineering

I certify that I have read this study and that in my opinion it conforms to acceptable standards of scholarly presentation and is fully adequate, in scope and quality, as a dissertation for the degree of Doctor of Philosophy.

Richard L. Francis

Dr. Richard L. Francis
Professor of Industrial and
Systems Engineering

I certify that I have read this study and that in my opinion it conforms to acceptable standards of scholarly presentation and is fully adequate, in scope and quality, as a dissertation for the degree of Doctor of Philosophy.

H Donald Ratliff

Dr. H. Donald Ratliff
Professor of Industrial and
Systems Engineering

I certify that I have read this study and that in my opinion it conforms to acceptable standards of scholarly presentation and is fully adequate, in scope and quality, as a dissertation for the degree of Doctor of Philosophy.

Timothy J. Lowe

Dr. Timothy J. Lowe
Associate Professor of Industrial
and Systems Engineering

I certify that I have read this study and that in my opinion it conforms to acceptable standards of scholarly presentation and is fully adequate, in scope and quality, as a dissertation for the degree of Doctor of Philosophy.

Gary J. Koehler

Dr. Gary J. Koehler
Associate Professor of Management

This dissertation was submitted to the Graduate Faculty of the College of Engineering and to the Graduate Council, and was accepted as partial fulfillment of the requirements for the degree of Doctor of Philosophy.

December 1978

Herbert A. Bevis

Dean, College of Engineering

Dean, Graduate School